AN ABSTRACT OF THE THESIS OF

Jose Cedeno for the degree of Master of Science in Computer Science presented on September 9 2010.

Title: Motivating Programmers Through Karma Systems

Abstract approved:

Carlos Jensen

Keeping FOSS developers motivated is a challenging problem, and their motivation levels can affect the team's productivity and satisfaction, leading to higher or lower productivity. Using reputation systems as a motivator has become the de-facto standard for many online communities, rewarding user's activity through badges of honor or achievement levels. Few open source software communities have successfully used a well-rounded reputation system to motivate developers, instead rewarding only one or a small set of activities. Very little research has been done in the area of using reputation systems to motivate people to increase their participation in open source software projects. This thesis studies Beaversource; a mix of code-hosting and social networking available to students, staff and faculty at Oregon State University, and our experiment with reputation systems as a means of motivating programmers. A survey was sent to 1,100+ Beaversource users at Oregon State University to gather demographics, and data on the use of Beaversource. One hundred users responded. Based on survey feedback, a reputation system was put in place. After the karma system was available to users, five students were interviewed to gather more information regarding their satisfaction with Beaversource and the karma system. Users reported that the reputation system worked to motivate them. Interviewees would like to see the karma system expanded with better icons and increased participation in the social networking side. Students also requested more flexibility in the karma system such as ability to affect each other's karma and give special badges to members of their project.

Motivating Programmers Through Karma Systems

by
Jose Cedeno

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented September 9, 2010
Commencement June 2011

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my Major Professor, Carlos Jensen for all his guidance and support during this project. I am thankful to the rest of my committee: Paul Paulson, Timothy Budd and Kipp Shearman for providing insightful feedback and comments regarding my project.

It is thanks to the support of my wife, Marilyn, that I was able to get through tough days at school. She was the one that kept me going and pointed me in the right direction instead of letting me take the easy route.

I appreciate all the help from my friends: Victor, Helen, Claire, Jen, Vignesh and everybody else in the HCI research group at OSU. Thank you guys for all the help and feedback you provided me during my thesis project.

A special thanks goes to my parents and siblings who always pushed me to keep going forward. I wouldn't be here if it wasn't thanks to my family support and love.

DEDICATION

I dedicate this thesis to the love of my life, my wife Marilyn, who never failed to be by my side during the tough times of this thesis project. I wouldn’t have been able to finish this project without your strong love and support at home.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF TABLES

1. Introduction

"Free open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of free open source software is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in" (OSI, 2010)

As defined by the Open Source Initiative (http://opensource.org/) quote above, free open source software can be thought of as a philosophy for writing programs or working in groups. The term free open source software (FOSS) was first applied to software in 1998 in a Usenet forum (FOSS, 2010). In 1998, Eric Raymond, a free open source advocate, helped Netscape plan their release of the Netscape browser as free open source software, that at the time caused quite a commotion as no other company had yet given code away (Revolution OS, 2001). This event and others to follow, as well as free open source advocates asking people to adopt the term, increased the use of free open source software (Teaching Open Source, 2010). FOSS gained even more momentum when the government of Peru used the term to pass a bill that would promote the use of FOSS across all its bodies (Pogue & Day, 2004).

More than 12 years later, since it's first use, free open source software, often referred to as FOSS, can be found in desktop software, including office productivity suites; web browsers; financial software; operating systems; web servers; and even in consumer electronics like smart phones, televisions and mp3 players.

FOSS communities can create work that rivals any proprietary solution. Wikipedia provides a great example of how a huge online community can come together to rival printed encyclopedias. Firefox, a web browser, and Apache, a web server, are two FOSS projects whose online contributors have come together to rival Microsoft products (Luther & Bruckman, 2008).

If FOSS projects are to compete with proprietary programs, they need a new influx of members and contributions. One method to accomplish this is to motivate programmers through reputation systems. Karma systems, also known, as reputation systems are a set of achievements, labels or points used to motivate users while showcasing their progress or contributions in a community. The purpose of this thesis project is to investigate how karma systems can be used to motivate community members. This study looked at how a karma system impacted a code community (Beaversource) at Oregon State University (OSU).

Before Beaversource (a mix of code-hosting and social networking) was set up at OSU, there was no place for students to practice and learn how to use FOSS tools like SVN, bug tracker, wiki or reading large code bases. A year and a half after it's launch, this community has grown to 1,250 users, 304 code projects and 83 groups. Thanks to Beaversource, students have better learning opportunities, but the site was lacking more user interaction with peers outside of their projects or close knit groups.

There were several factors behind the motivation of adding a karma system to Beaversource, one of them being to help users and projects identify good people to work with and collaborate in projects. Promoting good behavior in the system by rewarding users each time was a key factor during the karma system implementation. Another goal behind the karma system is to strike a balance across a variety of activities by rewarding all activities fairly. Last, I wanted to nudge users to explore the community site a bit more and use features that they may not have used otherwise. To achieve these goals, a set of research questions was formulated, as presented below.

RQ1: Can a karma system boost activity in the site?
RQ2: Is it possible to reward social activity through karma badges?
RQ3: Can a karma system reward multiple types of activities fairly?

To answer these questions, my research project consists of conducting a survey to gather information on the community and putting in place a karma system to address some of the problems identified by the survey results. To verify survey results and query student's satisfaction with the karma system, interviews were conducted.

It is my hope that this thesis can help employers identify students with FOSS skills and experience; aid universities so they don't have to reinvent the wheel; and help FOSS projects learn how to design reputation systems that reward multiple types of activities.

The rest of this thesis is organized as follows. Chapter 2 provides a literature overview of FOSS in education, FOSS communities and karma systems. Chapter 3 describes FOSS at Oregon State University and how Beaversource is used. Chapter 4 provides a methodology overview. Chapter 5 describes the karma implementation. Chapter 6 analyzes the survey results and interview responses. Chapter 7 discusses the findings of the study. Chapter 8 concludes this thesis.

1.1. Definitions

Below is a list of commonly used terms throughout this thesis.

Bug Tracker
Bug trackers are pieces of software used by almost all FOSS projects to keep track of bugs in their software.

Code-Hosting Site

Sites such as Google Code and SourceForge that can host a website, bug tracker, code repository, wiki and mailing list for coding projects.

Commit Access

Before developers can share their code to members using a central repository, they need to be granted permission or commit access.

CVS/SVN

A centralized source control management that developers can use to keep track of changes made to a project.

IRC

Internet Relay Chat or IRC is the main method of communication that many FOSS projects use. It is used as a broadcast group messaging service organized by channels. People connect to an IRC server using a client in their desktop machine that allows them to read messages posted to the chat room as well as post messages.

IRC Bot

As defined by Wikipedia, "An IRC bot is a set of scripts or an independent program that connects to an Internet Relay Chat as a client, and so appears to other IRC users as another user."

Karma Reputation or Reputation Systems

A set of achievements, labels or points used to motivate end users while showcasing their progress or contributions in a community.

Lurking

Idling, instead of participating in a discussion. Users may not be knowledgeable, feel confident or have time to engage in the discussion.

Proprietary Software

Programs usually developed by companies who are closed source and not open to the public for examination. People

Revision System

A tool that allows developers to keep track of the history and changes made to a project. People can provide messages each time they commit a new set of changes, as well as merge changes made between two people.

Sybil Attacks

This is where a user with multiple accounts in the system coordinates actions to increase his/her karma points.

2. Literature Review

This chapter will provide a review of the literature available on FOSS in education, FOSS communities and karma systems. First the use of FOSS in education will be described, followed by FOSS communities, their structure and the motivation behind them. Next will be a description and comparison between the FOSS and proprietary development environments. Next you'll find the issue of communication within projects and how it's currently addressed. The chapter will end with a description of reputation systems, and why they are important for FOSS projects.

2.1. FOSS in Education

Moodle (http://moodle.org/) and Mahara (http://mahara.org/about) are two examples of how FOSS can be used to support educational institutions (K-12 as well as higher education) without students and faculty realizing they are using FOSS. Moodle allows educational institutions to host their online courses and websites and it's actively developed by people across the globe. Mahara was initially funded by New Zealand's Tertiary Education Comission's E-learning Collaborative Development Fund to provide an ePortfolio system that was learner centered. Moodle and Mahara can also work together to empower teachers to provide online courses and information to students, as well as showcase students' projects and progress throughout the years.

Besides providing backend support, FOSS is also used in academia to enhance student's learning by providing them with real world experience and exposing them to new challenges, allowing them to grow. One example of this is the Sahana project.

Sahana (http://sahanafoundation.org/), a FOSS disaster management system, solved common coordination problems like finding missing people, managing volunteers and tracking camps. It was developed in 2004 by Sri Lankan volunteers to aid the volunteers during the 2004 Tsunami. Ellis and Morelli (2006) from Trinity College reported on their experiences using Sahana as an experiment to better the undergraduate Computer Science curriculum (Ye & Kishida, 2003). They explored the use of Sahana to teach free open source software development while motivating students to make a social contribution to the community, by donating their expertise, time and efforts in making the software better.

The use of Humanitarian Free Open Source Software (HFOSS), such as Sahana, to strengthen the CS curriculum of Trinity College was successful,

> *"Like most CS students, we had never been contributors to FOSS, and working on Sahana was something of a revelation. Typical programming assignments begin with a blank screen. But in this context, there was a huge repository of pre-existing code, so writing a new page almost inevitably involved modification of an old one. At first surprising and later gratifying ..."*

Students were able to learn more about documentation and why it was important; how to write better self-documenting code; stress and complexity of real-world development environments, and built professional contacts, by having to travel and work with developers from companies such as Google and Microsoft. Due to the success of the program, Trinity College along with Wesleyan University, Connecticut College and Oregon State University, formed the Humanitarian FOSS Project (http://hfoss.org/) to create a community of academic computing departments, IT corporations, local and global humanitarian volunteers, and community organizations dedicated to help the spread the use and development of FOSS to better humanity.

The Association for Computing Machinery (ACM) computer curricula recommendations, based on surveys and feedback received from industry found that students who had just graduated with a Baccalaureate degree needed more experience in: testing, debugging and bug tracking, code reviews, release management, source control principles, team work and, experience working with large poorly documented pieces of code (ACM, 2008). The IT industry began to take special note of students who had FOSS experience and had done internships with companies in the industry, as well as students who had done a project for their class work or research.

The IT industry recommendations to the ACM and their impact are seen throughout university curricula where students are exposed to FOSS methodology and tools. For example, OSU's Open Source Lab does a great job of giving students real world experience with system administration of servers that provide support to FOSS projects (About OSU OSL, 2010). At the same time, students also get to write code for FOSS projects, if funding allows. The University College Dublin, in Ireland, has a similar FOSS lab, where they provide students, staff, faculty and enthusiasts with a place to collaborate and develop FOSS by providing them with backup and code-hosting services (OSL University College Dublin, 2010). Oregon State University has the Open Source Education Lab (OSEL), a group with the sole purpose of helping students at OSU get involved with FOSS (OSU OSEL, 2010). Stanford's Open Source Lab focuses on making sure that the knowledge people produce and consume at Stanford is available to the public through mechanisms such as: open workshops, open access publishing and software development (Stanford Open Source, 2010). Harvard Forge provides students, faculty and staff at Harvard with code-hosting services, such as wiki, bug tracker and code repository. Clarkson University has an Open Source Institute that provides students with equipment and support for them to work on FOSS projects, as well as provide a place for them to experiment and learn.

FOSS tools and their use are also making it to the classroom's curricula. For example, at OSU students are exposed to tools such as wikis and revision control (https://secure.engr.oregonstate.edu/classes/eecs/spring2010/cs419-003/index.php/Main/HomePage) in their 300 and 400 level CS classes (https://secure.engr.oregonstate.edu/wiki/CS-411/index.php/Main/HomePage). These tools are fundamental and basic when it comes to collaborating in projects and doing code development. Students in engineering and IT related fields should be exposed to them as early as possible and be extremely fluent and comfortable with them by the time they graduate.

In 2007, Su and Jodis did an experiment where they provided students with an integrated team software development environment that was composed of Eclipse, an Integrated Development Environment (a program to edit code with), project build tools, SVN, bug tracker, continuous integration tools (monitors changes and compiles code) and a mailing list (Su et al., 2007). Students were required to install the software in their computers and instructors configured the software in servers. This made for an easier introduction to FOSS for the students. Instructors found that students were better prepared and could mimic standard industrial development environments using FOSS. One challenge identified by Su and Jodis was the need to develop better documentation and guidelines to lower the learning curve for students.

2.2. FOSS Communities

Free open source software (FOSS) projects depend greatly on the people using the software. This group of users form a community that contributes to each FOSS project by helping with testing, reporting bugs, providing support, helping with documentation, promoting projects by talking about them and making donations. Most FOSS projects begin with one or a handful of developers working on an idea to "scratch an itch," as explained by Eric Raymond (Raymond, 1992). "Scratching an itch" refers to the practice of addressing issues or problems experienced by the programmers themselves, thus "scratching an itch". Even though end users do not help with code, they play a key role in FOSS development.

Ye and Kishida, (2003), explain that FOSS projects form a community around them with a hierarchy that resembles that of an onion (Figure 2.1) (Crowston & Howison, 2005; Ye & Kishida, 2003). At the center of the community are the project leaders that in small projects could be just one person. In larger projects, the center could be comprised of a release manager, project leaders or a committee. This layer is often smaller than the outside layers. The project leaders are in charge of determining the direction of the project, releasing new versions, freezing code to fix bugs, and doing community outreach.

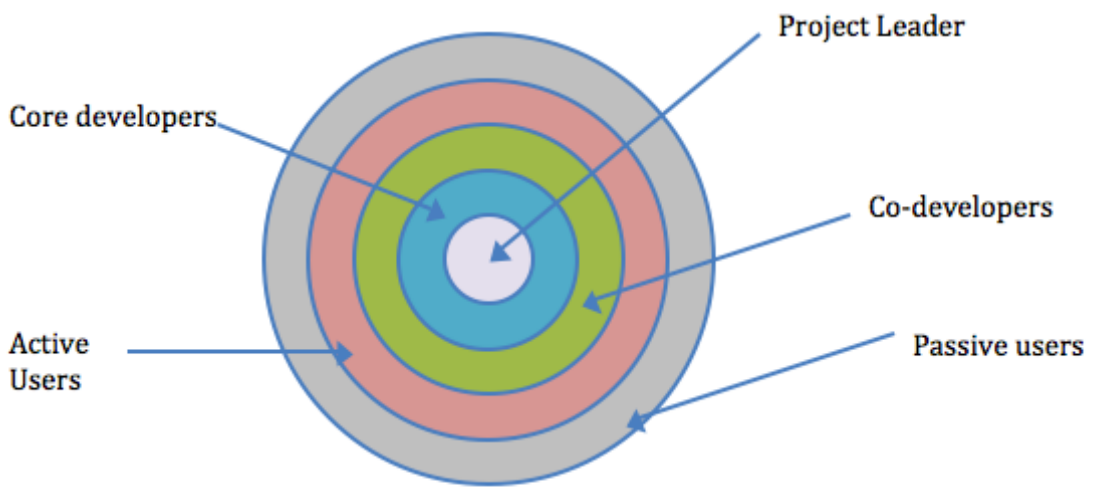


Figure 2.1 FOSS Community Layers (Crowston & Howison, 2005)

Outside of the project leader layer is the core developer group. It consisted of people with commit access to the project. They are a set of experienced developers who close bugs, review patches submitted by other community members and co-developers. Core developers have a very detailed understanding of the code and are able to work on core-features. This group also provides feedback to the project leaders on where the project should be going.

Co-developers are a set of community members who don't have commit access to the project. They actively work on small features or bugs that affect their daily usage of the program and submit their code in the form of patches to core-developers. They have a good understanding of select areas of the code, but may not be familiar with the full code base. For example, the bulk of contributions to the Linux Kernel are made by non-core developers who submit a one-time patch (http://lwn.net/Articles/222773/). Co-developers also help with documentation and end user support, to take the load off core-developers.

The active users layer is made up of users who are tech-savvy. They actively participate in the online discussions in forums, mailing lists and IRC channels. They begin by asking simple questions, but as time passes and they gain more knowledge and pay it forward by taking care of easier support tasks, thus taking the load off co-developers. Users in this category help test beta and alpha releases of software and help test out new features by submitting bugs. Due to the size of this group, they are able to use the software in conjunction with a great variety of hardware combinations that developers typically don't have access to, or they help with software configurations that developers wouldn't have imagined.

The last layer, passive users, is made of users who just download the software and don't participate much in the community. This group also contains the 'lurkers,' or people who join mailing lists and online forums, but stay in the back listening, but never participating. There is no way to estimate the size of this layer because passive users are able to download the software from many websites.

This analogy of layers is still commonly accepted by the FOSS community and has been observed by many other researchers, such as Crowston and Howison (Crowston & Howison, 2005), who found that the structure of a FOSS project can be strictly centralized as in Figure 2.1 or de-centralized in which there are multiple sections that operate separately as separate instances of Figure 2.1. Each layer of the onion was larger than the layer inside it. FOSS communities increase the number of people in each layer by having users from the outer layers move into inner layers. For example, a person may begin as a passive user, but then transition into the active users groups by increasing his/her participation. This user then starts submitting patches and becomes a co-developer, and after contributing to the project for some time, and proving his or her worth, this co-developer receives the role core developer

A healthy influx of new users can help ensure that a project lives on even as core developers leave a project. End users can become more knowledgeable about the project and a few of them start joining the group of co-developers. This is why understanding the motivation behind FOSS developers, and how end users can be enticed to increase their contributions and move into the inner layers, is critical for the survival of FOSS projects.

2.3. FOSS Development Environment

Before we can understand what motivates FOSS developers, we need to establish what it is that they do on a daily basis, what tools they use, and how their environment differs from a regular proprietary development.

FOSS development has changed over the last three decades, and there's no one-size-fits-all description for FOSS projects or their environments (Revolution OS, 2001; Raymond, 1992). There are big companies, such as IBM, working on free open source software for their products (http://www.ibm.com/developerworks/opensource/), and they support kernel developers to make sure Linux runs well on their hardware. Then there are big to mid-size hobby projects, such as cyanogenmod (http://www.cyanogenmod.com/), where the core developers work on the project during nights and weekends. There are also thousands of small projects, on sites such as Google Code (http://code.google.com/), SourceForge (http://sourceforge.net/) and GitHub (http://github.com/), made up of one or two developers working on their ideas. The development of a FOSS project can be centralized; people working on FOSS in the same building, or it can be distributed; people on different continents working on the same project. Even though some core developers may work in a central location, the tools used for development and communication are de-centralized by nature to make it easier for co-developers to work remotely. It is this requirement that promotes the use of tools such as: IRC channels, wikis and mailing lists. This distributed model provides some flexibility not usually found in commercial software development. Developers can work in the comfort of their home, while on a plane, or in the middle of the night. (Ellis et al., 2007)

Most projects have a web portal where users and developers can check on the latest activity of the project. For example, SourceForge (Sourceforge 2010; Halloran T. J., 2002), provides a dashboard page that gives developers a general overview of what changes have been made since the last time they visited the site. FOSS programmers have to keep track of many things at once, such as patches from co-developers, updates from other projects, and active users' feedback. Having a dashboard-like system makes it easy to view the activity of a project in the last week, 24-hours or month, which makes it easy for them to get an overview of what's going on project-wide.

Developers use source code control systems, such as CVS, SVN or Git, to generate patches and commits, and to share their code contributions with the rest of the team. Along with a source code control system comes a code viewer such as: ViewCVS, Cgit, CVSWeb, and WebSVN, that provide a Graphical User Interface (GUI) to show changes in the code.

2.3.1. FOSS vs. Proprietary world

Motivation is an obvious difference between the FOSS and proprietary development world. At work, your boss tells you what project or features you have to work on and the motivation is most likely monetary. On the other hand, FOSS developers typically volunteer their time, and are more likely to spend their evenings on something that they enjoy doing. Some people are proprietary developers during the day and FOSS developers on nights or weekends (Crowston & Howison, 2005; Ye & Kishida, 2003).

Proprietary developers applied for a job and proved their qualifications through their resumes and the interview process. FOSS developers proved their worth by submitting patches and participating in the community, catching the attention of others. In FOSS communities, your educational background and experience don't matter as much. You can be a teenager in high school, a college student or somebody who has been working in industry for decades and receive the same treatment (Raymond, 1992; Revolution OS, 2001).

The set of tools that FOSS developers and proprietary developers use to write code are similar. One main difference is due to the different communication needs. Proprietary developers often work in cubes and have white-boards at meetings and face-to-face technical discussions. Instead of having to use IM people can just walk over and ask a question. In proprietary development, roles are typically formally defined; the development team, the UI design team, the testing team, etc., and these need to collaborate This means that proprietary developers depend heavily on calendar and email applications to manage meetings (Crowston & Howison, 2005). In contrast, FOSS developers usually adopt multiple roles in their projects, and developers often have ad-hock meetings in IRC or have to schedule meetings at random times of the day so that everybody can be present.

2.4. Communication within projects

FOSS projects are found all over the globe, as are developers and project members, so effective communication is a basic requirement. Communication tools and their use by FOSS projects can shed light on the project structure and hierarchy, as well as affect design and development decisions.

2.4.1. IRC

Internet Relay Chat, or IRC, is the main method of communication for many FOSS projects. FOSS projects often create a communication channel and designate people to help moderate it. Big FOSS projects such as Mozilla and Gnome have their own servers, while others use shared servers such as Freenode to communicate. There are also FOSS projects that have an IRC channels for developers and for users, as well as channels for languages other than English (Teaching OSS, 2010; Open Source as Programming Experience for College Students, 2002; Halloran T. J., 2002).

IRC channels provide a great avenue for asynchronous communication. It is not unusual for people to keep their IRC client open all day. This can lead to the response time seeming extremely slow. On the other hand, during heavy traffic hours it can be almost impossible to keep track of all the communication as four or more simultaneous conversations can be going on at the same time.

As pointed out by the Kernel Newbies community, IRC is a great way to get started with FOSS development, learn about a new project, start discussions, and brainstorm (Kernel Newbies, 2010). Lurking or idling in a IRC channel helps new members get to know the developers, the active members in the community, and to learn more about the current project, by reading the various questions and discussions in the channel. Eventually some of the lurking members may become active members by joining discussions or helping newer members, who have come asking for help.

Some FOSS projects log and publish IRC conversations on the web for both future reference and to allow members to catch up on conversations they may have missed. IRC bots are common in many IRC channels and provide automated functions such as: answers to simple questions, bookmarks to FAQs and documentation, logging functionality, and more advanced tasks, such as granting users extra privileges when they first enter the channel. These extra functions allow the user to moderate discussions, update the topic of the channel, ban people and many more administrative type tasks. Not all channels welcome bots since the scripts that make up a bot can also be used for spamming users. These malicious bots can create unwanted increased traffic on some networks (Teaching Open Source, 2010; IRC, 2010).

Using IRC comes with an unexpected learning curve for new users. “Can I ask a question,” “Is anybody here,” “Can someone help me” are common questions asked by new users not realizing they are breaking a common IRC channel etiquette. This etiquette and other rules are explained in the channel topic, but the topic scrolls by very fast when the users first join the channel. Depending on the tolerance level of the IRC channel, users may be taught the rules, or get kicked out of the channel for breaking the rules. This can put off new users who may find these interactions rude and unfriendly (Teaching Open Source, 2010).

2.4.2. Mailing Lists

Often used in addition to IRC, or as a replacement. Code hosting sites such as SourceForge, Launchpad and GoogleCode offer mailing lists for the projects they host (SourceForge, 2010; Google Project Hosting, 2010). Mailing lists are a great tool for asynchronous communication in FOSS projects.

When it comes to mailing list software, there are various options: Mailman, Google Groups, Majordomo, and Ezmlm. Some FOSS projects have one mailing list for developers and a separate one for end-user support/help requests. Bigger projects can have language-specific support mailing lists. Mailing lists are great places for discussions and some projects, such as the Linux Kernel heavily encourage new developers to send patches (bug fixes to a program) to the mailing list to receive feedback and review (Halloran T. J., 2002). Some discussions can be very heated and generate a lot of comment from developers and end-users. Passionate users can have heated discussions, often referred to as "flame wars'' that can span hundreds of messages. A FOSS developer can easily get hundreds to thousands of emails per day, thus the sheer volume of emails may lead FOSS developers to opt for a separate email for mailing lists or filter to parse through the mailing list traffic (Teaching Open Source, 2010).

As noted above, it is not uncommon for users to lurk in mailing lists. Some of them join mailing lists to receive notifications and choose to respond only if the topic "scratches an itch," or they are able to help with the project in some way (Raymond, 1992). Some FOSS mailing lists and their logs are open to the public in order to encourage participation, discussions and collaboration with the community regarding new features, bugs and implementation. Some mailing lists are private and open only to a select group of core developers. For example, the Apache project discusses security issues on a private mailing list instead of a public one (Halloran T. J., 2002; Teaching Open Source, 2010).

Some mailing list software such as Google Groups can be hybrid and act as both a forum and mailing list. This provides greater flexibility to those who may want to participate in a discussion, but don't want to sign up for mailing lists.

2.4.3. Blogs

Blogs are commonly used by FOSS projects to reach the community and provide them with news and updates for the project. Depending on the size of the project, the blog may be managed by either a developer or a community manager.

Besides a project blog, many FOSS developers have a separate personal blog that acts as a gateway for social interaction with community members and peer developers. Aside from personal updates, FOSS developers may mention upcoming project developments and FOSS bugs/tasks that they are working on. The blog entries made by FOSS developers are usually professional, in-depth and with enough detail to get feedback and foster discussion with community members (Teaching Open Source, 2010). Commonly, developers will post screenshots of updates in order to get feedback during the early stages of the design process. Eventually these screenshots and postings are turned into documentation in the form of wiki pages.

Big FOSS projects, such as PHP, Phython, and Ubuntu, aggregate the blog posts of their developers using a software program called Planet. Planet was first developed for Gnome and Debian and creates a unified blog page sorting the posts from most recent to least (Planet, 2005).

2.4.4. Wiki

No other piece of software embodies community work and collaborative web development like wikis do. Wiki is probably most well known for Wikipedia, a set of online encyclopedias which viewers can contribute content/edits/references. As described by TrackWiki, "the main goal of the wiki is to make editing text easier and encourage people to contribute and annotate text content for a project" (Trac Wiki, 2010). Besides enabling fast edit/revise cycles, wikis provide an easy to use revision system. Viewers can use this revision system to compare changes made to an article, revert changes, and track page edits to the user who made the changes. Wikis are flexible and don't impose any structure on content creators.

MediaWiki is the most popular wiki software because it's easy to setup, scalable, and powers the most visible and successful wiki, Wikipedia. There are other wiki engines as well, including Redmine, TracWiki, and PmWiki. Wiki engines vary on the language that they are written in such as PHP, Python, Ruby and others; as well as how data is stored, either by database or file system. None of these differences affect the end user as much as the wiki markup language used. In order to write paragraphs, unordered/ordered lists, tables, and links, the end user must memorize the markup language or rules used by the wiki (How does Mediawiki work?, 2010).

FOSS projects use wikis extensively for documentation, status reports, brainstorming, meeting minutes, and conversation logs. They allow FOSS projects to harness the power of its community. In small FOSS projects, developers are usually the ones to write wiki pages. As the project becomes more popular, developers switch to writing more of the barebones-type wiki page, describing an upgrade or installation process, then advanced end-users write more of the documentation and article details. Many big FOSS projects have dedicated documentation teams that write wiki articles; update existing documentation pages and review contributions made by other community members after checking for accuracy and correctness (Teaching Open Source, 2010) .

As a method of communication, wikis affect FOSS projects because their content is usually more permanent, with fewer changes as in mailing lists, IRC or blogs. Even though wikis make content more accessible to users, the quality of its content depends greatly on how involved the FOSS community is with contributing and checking the accuracy of the information (Riehle, 2006). A wiki, through its revision history, can easily show the people most involved in documentation, as well as how active or stale the wiki content is. Participating in documentation is one of the easiest ways for new members to get involved with an FOSS project (Luther & Bruckman, 2008).

2.4.5. Bug Tracker

A bug tracker is usually a website, accessible to all users in the community, where any community member can post a bug, request a new feature or a task that needs to be completed. Bug trackers are an important channel of communication because each bug acts as a mini mailing list, were members post comments and hold discussions. Some bugs are easily resolved, while other bugs can take dozens or hundreds of comments before they are closed. Code-hosting sites like SourceForge and GoogleCode offer bug tracking as one of the essential tools for developers. There are a great variety of bug trackers available; some examples are: Mantis, Trac, Redmine, and GNATS (Halloran T. J., 2002; Teaching Open Source, 2010).

Most bugs have the following fields: title or summary, description, priority, status, owner, reporter and attachments. The only required fields when submitting a bug are the summary and description of the bug itself. Most bug tracking systems will automatically set a default status and priority to the bug and set the name of the user who reported the bug. The status of a bug can be: new, assigned, resolved or closed. The attachments section of a bug allows a user to submit logs, screenshots or other relevant information regarding the bug. Power users can take advantage of the attachment field and submit a patch or propose a fix to a bug. This allows non-core developers, without commit access to a project, to submit patches and code to a community. Core developers post comments in the bug providing feedback to the patch. They eventually close the bug and include the fix, if it is appropriate (Halloran T. J., 2002).

Bug trackers become a to-do list for developers, but unfortunately the bug list can often be disorganized. Part of the reason for the disorganization is that many new users post 'me too' responses to bugs, making useful information hard to find (Ko & Chilana, 2010). It is often the case that there are more bugs than time to fix these, and thus getting the attention of a developer who can fix a bug is important for the person who reports the bug. FOSS developers have pointed out that there's a big gap in the knowledge of bug reporters. Some bug reporters are rude or sarcastic in comments or don't provide enough information with their report. These hinder their chance of having their bug fixed. On the other hand, more experienced users, or community members who have good reputations, will draw attention to a bug because their names are associated with the bug. Sometimes bugs will be used to argue the case for a feature, or to debate on the importance of a use case or scenario (Ko & Chilana, 2010).

In successful FOSS projects, the group of people fixing bugs is larger than the group of core developers, and the group of people reporting bugs is larger still (Crowston & Howison, 2005). This shows that bug tracking is a crucial piece of communication in the FOSS ecosphere. The people fixing the bugs must understand what the people reporting the bugs are trying to communicate. A good bug report usually contains the answer to three questions: what are the steps to reproduce the problem, what did you expect to see, and what did you see instead? After a bug has been reported, it is usually assigned to a developer in the project. The bug then becomes like a hot-potato. The developer assigned to the bug must fix the bug, provide comments, or reassign the bug to someone else. The person who closes the bug should be the same person who reported it. This helps ensure that the bug fix is tested within the same conditions as it was reported (Spolsky, 2010).

2.4.6. Social Networks

Social networking sites such as Twitter and Facebook, known as great tools for keeping in touch with family and friends, are starting to creep into FOSS projects as well. Many times community managers and developers will use their facebook and/or twitter accounts to send updates, as well as links to news/blog posts, to their followers. This becomes an instant and non-obtrusive way for FOSS core members to reach the community. FOSS Developers use social networking sites to rant, ask for immediate feedback and share thoughts from other community members, in addition to keeping tabs on developers' projects (Ohira et al., 2005).

2.5. Karma/Reputation Systems

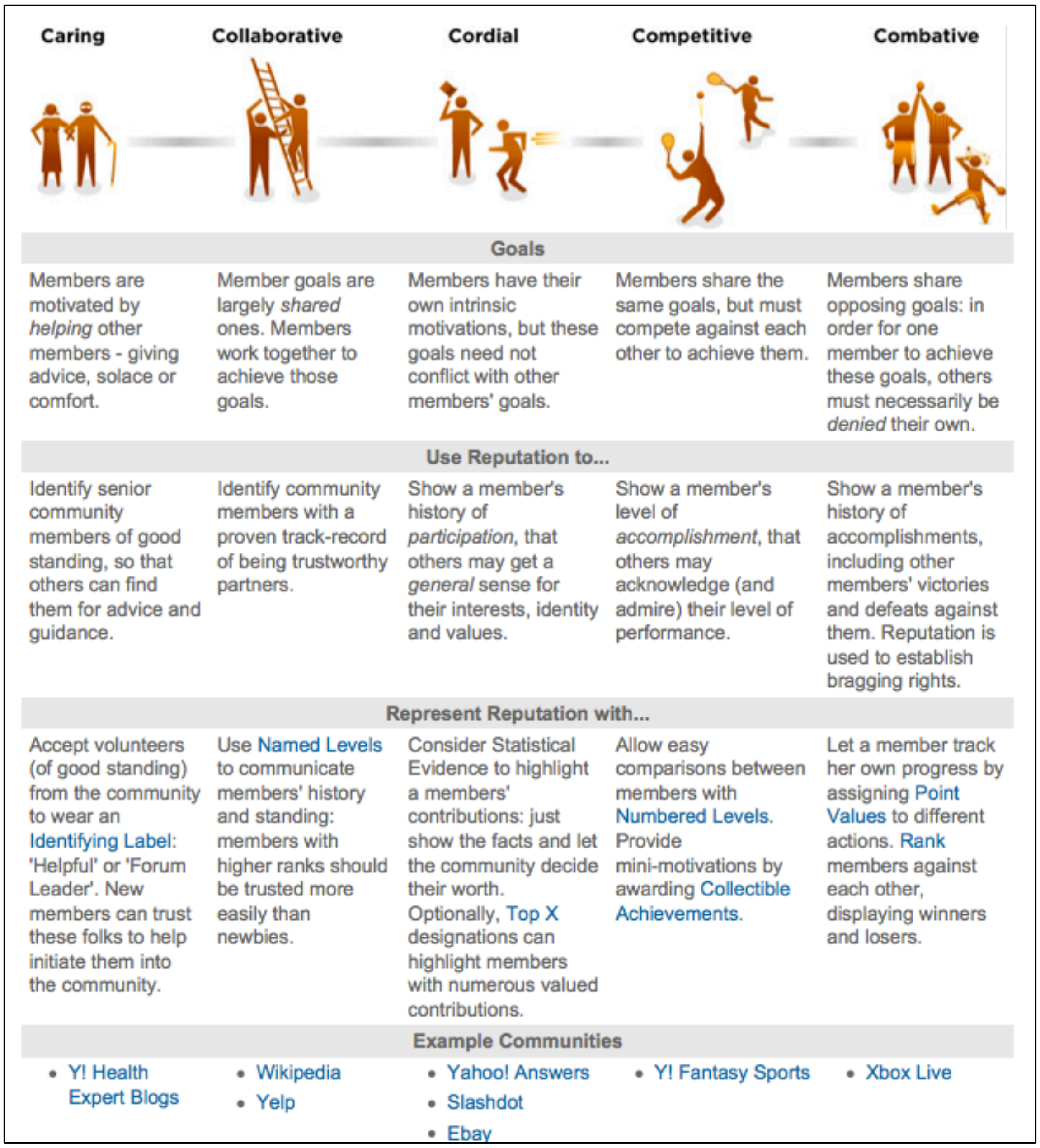
Reputation systems, also known as karma systems, use metadata regarding a user's online activity, including: number of posts/photos/friends/files/and other online data; helpfulness of the posts; length of membership; and many other metrics. These metrics are then used to give points or labels to the user. The user can then be ranked or compared against other online members to assess activity (Harper et al., 2007).

Ebay's successes during the early parts of 2000, was due to their reputation system, known as the Feedback Forum. After each transaction, the buyer and seller could rate each other and leave a comment. Due to the success of reputation systems in sites such as Ebay, karma systems started to pop up in other areas of the Internet.

Before reporting why reputation systems help motivate and build trust among strangers, it is important to look at how people build trust outside of the Internet. Axelrod studied and wrote on the evolution of cooperation, provided insight into human interaction over long periods of time (Axelrod & Hamilton, 1981). In his 1984 paper, he observed that people who interact over a long period of time take advantage of previous interactions, to learn each others abilities and dispositions. The expectation of reciprocity, or retaliation, in future interactions worked as a motivator for good behavior. Axelrod coined the term, "shadow of the future", to describe the expectation that people will take into account previous experiences in the present.

When two strangers interact online they can be tempted to not hold their end of the bargain (Resnick et al., 2000; Axelrod & Hamilton, 1981). Reputation systems provide feedback that helps build the trust historically generated over long-term relationships. Buyers and sellers can leverage their online reputations to get higher prices or better services in a community (Resnick et al., 2000).

Karma systems have become a popular method to motivate participation in online communities. The Yahoo Developers' Network (YDN) patterns library, focuses on reusable user interface patterns. One set of the YDN patterns provides an excellent insight on reputation systems. These the reputation patterns have advantages, disadvantages and their own biases which help decide when it's appropriate to use them (YDN Reputation Patterns, 2010).



Caring	Collaborative	Cordial	Competitive	Combative
Goals				
Members are motivated by *helping* other members - giving advice, solace or comfort.	Member goals are largely *shared* ones. Members work together to achieve those goals.	Members have their own intrinsic motivations, but these goals need not conflict with other members' goals.	Members share the same goals, but must compete against each other to achieve them.	Members share opposing goals: in order for one member to achieve these goals, others must necessarily be *denied* their own.
Use Reputation to...				
Identify senior community members of good standing, so that others can find them for advice and guidance.	Identify community members with a proven track-record of being trustworthy partners.	Show a member's history of *participation*, that others may get a *general* sense for their interests, identity and values.	Show a member's level of *accomplishment*, that others may acknowledge (and admire) their level of performance.	Show a member's history of accomplishments, including other members' victories and defeats against them. Reputation is used to establish bragging rights.
Represent Reputation with...				
Accept volunteers (of good standing) from the community to wear an Identifying Label: 'Helpful' or 'Forum Leader'. New members can trust these folks to help initiate them into the community.	Use Named Levels to communicate members' history and standing: members with higher ranks should be trusted more easily than newbies.	Consider Statistical Evidence to highlight a members' contributions: just show the facts and let the community decide their worth. Optionally, Top X designations can highlight members with numerous valued contributions.	Allow easy comparisons between members with Numbered Levels. Provide mini-motivations by awarding Collectible Achievements.	Let a member track her own progress by assigning Point Values to different actions. Rank members against each other, displaying winners and losers.
Example Communities				
• Y! Health Expert Blogs	• Wikipedia • Yelp	• Yahoo! Answers • Slashdot • Ebay	• Y! Fantasy Sports	• Xbox Live

Figure 2.2 YDN's reputation and competitiveness matching (YDN Reputation Patterns, 2010)

2.5.1. Competitive Spectrum

As pointed out by YDN, the first element to identify before putting a karma system in place is the competitive spectrum (see Figure 2.2),

"The degree of competitiveness of a community depends on the individual goals of community members, the actions they engage in, and

> *to what degree inter-person comparisons or contests are desired. Articulating the community's competitiveness can help the designer of a reputation system determine which specific reputation patterns to employ."*

If a karma system is introduced into a community without thought it may cause more harm than good. The chart in Figure 2.2, provided by YDN, establishes various levels of competitiveness and what reputation pattern best aligns with the competitive needs of that community (YDN Reputation Patterns, 2010).

As pointed out by a blog post in XOXCO entitled "I love my chicken wire mommy", people who design online communities must invent methods to continually reward and motivate the behavior they want to see in their community (I love my chicken wire mommy, 2009). In his post, Ben Brown describes how Consumating.com's reputation system caused havoc in its online community and failed to motivate the behavior that organizers wanted to see. Some community members, who were very competitive, obsessed over acquiring points instead of providing value to the community. Consumating.com allowed people to take points away and thus new members could have higher karma than older members. Consumating.com displayed charts and information regarding a person's karma, but they were not very useful.

In a blog post from the HorsePigCow, a blog dedicated to social networks and their inner workings, Miss Rogue, the owner of the blog, stated, "The health of a community is the gauge of where various qualitative and quantitative metrics lie in relation to the goals you set." This blog emphasizes the importance of identifying the various qualities and behaviors that are to be encouraged in a community. The next step is to define a list of metrics to measure these goals (Metrics for Healthy Communities, 2007).

2.5.2. Named Levels

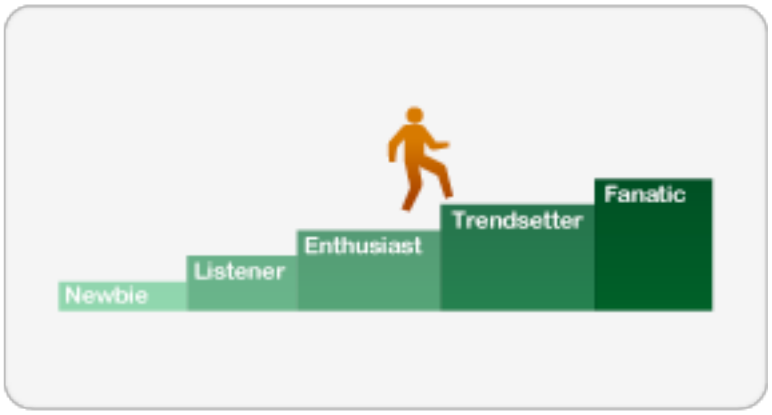


Figure 2.3 Named Levels (YDN Reputation Patterns, 2010).

The next reputation pattern as described by YDN, is "named levels" (see Figure 2.3), where there is "a family of reputation levels on a progressive continuum. Each level is higher than the one before it" (Metrics for Healthy Communities, 2007). The labels are designed to make the reputation system more attractive to community members. This type of reputation system is best suited to help users identify who has more experience in a community, as well as higher quality contributions. Named levels are meant for communities who are slightly competitive, but not highly competitive. The progressive continuum of named levels makes it easy for users to keep track of their progression to the next level that they can achieve. Some recommendations for this reputation pattern are:

- Use easy to understand names. Clarity is more important than fun and cryptic names that new members may not understand;
- Use fun names, for the levels related to themes of your community; and
- Add a level at the high end to motivate users with high karma scores to achieve a level they have not yet reached.

The next reputation pattern is "numbered labels". Some people may find labels to be offensive. New community members may misunderstand the meaning of the named levels and can be thrown off by them. Some examples of named levels are: Yahoo's EU Sports forums and World of Warcraft. Yahoo's EU Sports uses gold, silver and bronze in their forums to indentify the quality of the member's contribution to the game. World of Warcraft uses named levels (revered, honored, friendly and neutral) to specify reputation between clans.

2.5.3. Numbered Levels

The numbered levels reputation pattern is similar to the named levels pattern. The main difference is that instead of having names in a progressive scale, a set of numbers is used. Each level is referred to by its number, making comparisons between levels easier (Metrics for Healthy Communities, 2007). This pattern is suited for communities who are interested in:

- Users tracking their own growth in the community;
- Having a large scale such as World of Warcraft that has 70 levels;
- Making comparisons between levels easy; and
- Trying to encourage competition between community members.

YDN recommends keeping the scale to no larger than 10 numbers to start, with higher levels to be added later on if needed. During user testing YDN found that some users reacted very strongly towards numbered levels, and stated that they felt as if graded, and they found the system to be cold and impersonal. This indicates that numbered labels must be applied to a community with care. An example of a successful community using numbered levels is World of Warcraft, where a user keeps track of his or her progress through the levels, and as a reward for achieving higher levels, the user has access to more exclusive content within the game.

Identifying labels or badges is the next pattern mentioned by YDN, is a set of reputation labels that, unlike named/numbered levels, are not sequential. The labels in this pattern are meant to help identify and reward community members for specific behaviors or qualities desired by the community, such as

> *"perhaps they've excelled at one particular skill that the community values; perhaps they are official representatives for the community or an affiliated organization; perhaps they have volunteered to be a helpful resource for others in the community"* (Metrics for Healthy Communities, 2007).

It is also common for this type of reputation system to reward points to a user based on actions that other community members perform towards the user. For example, if a user clicks a thumbs up to rate a comment, the author of the comment should get points for that action. This reputation pattern is best suited for identifying desirable behaviors; allowing members to volunteer for roles and take on responsibilities within the community, reflecting that the user has been validated by the community as a trusted 3rd party; and any community without regards on how competitive they are. This pattern is used by FOSS forum software such as Mybb (http://www.mybb.com/) and PHPBB (http://www.phpbb.com/), and thus by many of the FOSS projects using this software. When a community is using this pattern, YDN recommends to :

- Allow people to have more than one label, for example: developer and documentation team; and
- Allow users to apply for the identifying labels and to make the labels more exclusive.

Yelp, an online community where people rate restaurants and other venues, has an 'Elite Squad' label to help identify the most active and influential users. Yahoo Answers, an online community where people can post questions and get answers, provides a top-contributor badge to visually highlight members who actively contribute in discussions.

2.5.4. Points Reputation Pattern

The points reputation pattern's goal is to keep track of the cumulative number of points a community member has earned. The points are earned by performing specific actions within the community. This pattern is best suited for communities that are highly competitive, and where members want to easily compare their progress against peers. YDN recommends using points as a supplemental reputation pattern, where the points are not the main indicator of a user's reputation. For example, using: Yellow Belt 3 instead of Level 3 for a user's reputation in a martial arts community. As pointed out by YDN, the focus of points is to reward performance rather than activities (in a game site, a user would earn a point for winning a game rather than playing a game). The one exception to this recommendation is when a user performs an activity for the first time and he/she gets a reward. Some examples of communities using point-based reputation systems are: Yahoo Answers, Xbox Live Gamerscore, eBay Feedback Score, and Amazon Seller Feedback.

2.5.5. Collectable Achievements

As YDN mentions, "Collectable Achievements may seem silly or trivial, but they can have an addictive quality that may compel your users to explore parts of your offering that otherwise might not appeal to them" (YDN Reputation Patterns, 2010). It is recommended that when setting up this type of reputation system, you design attractive and exclusive badges and trophies, and making it easy for users to save and display them. Another YDN recommendation is to lock some of the badges so that they are only available after you have earned the more basic collectible achievements. Just as with the points system, it is helpful to assign collectible achievements for first time achievements. The key to collectible achievements is to make them exclusive. Basically a small secondary set of badges should be easy to achieve for new members, but not too many. An example of successful collectible badges is XBox Live, which awards gamers with exclusive badges for completing tasks within games.

2.5.6. Ranking Reputation Pattern

The ranking reputation pattern as described by YDN as split into two smaller patterns: leaderboard and top x. The leaderboard is meant to display a fixed ranking of top competitive community members, such as a top10 (YDN Reputation Patterns, 2010). The ranking karma pattern is designed to make player-to-player comparisons relatively easy within highly competitive communities. Ranking reputations are not meant for use with community-based activities that are not competitive in nature. It is useful to provide multiple leaderboards such as: top competitors of today, this week, this month, year and of all time. Leaderboards are not for everyone though. The following comment was posted by ShadeHunter in an online gaming forum,

> *"I mentioned this in another thread, but it bears repeating. Leaderboards don't work. Even if you get past the Timmies and the cheaters, the people at the top of the boards are just the ones who have the most time on the game. If you didn't get GoW on launch day and you aren't willing to spend 40 hours a week online, you won't ever be at the top of the board. I wish they'd get over the whole ranked/unranked thing and just let us play the game"* (Problems and Cheaters Curb Stomp Emergence Day, 2009).

A similar dissatisfaction was present with Amazon's classic leaderboard, as pointed out by Joshua Porter in a blog post where he talks about Amazon's #1 book reviewer, "Klausner is apparently trying to game the system so she keeps her position. In a world where building social tools like this is becoming more common every day, Klausner is diluting the value of her reviews just for personal gain. While nobody is going to get too upset over less-than-helpful reviews, the larger, longer effect is that if she's merely writing them to keep her spot, she's not writing them for the right reason. Amazon's social design should incentivize her to write valuable reviews, not allow her to write them without value" (Porter, 2010).

The second ranking reputation pattern, top x, "groups contributors numerically by performance, and acknowledge top performers for their superior achievements. Top 10, 50 and 100 are some commonly used groupings" (YDN Reputation Patterns, 2010). This pattern is recommended when communities want to distinguish top contributors and help them serve as examples for people in the lower ends of the top x scale. By using top x, you are trying to motivate people, who are good contributors, to model the behavior of heavier contributor members. YDN suggests that this pattern is made specific by providing separate top x buckets/placeholders for different categories of a community. For example, top 5 documentation contributor and top 5 patch reviewer means there are 5 placeholders or positions (1-5) in two categories: patch reviewer and documentation. Good examples of top x reputation systems are Amazon's Top Reviewers program and FBI's 10 Most Wanted list.

Yahoo Answers, for example, is an online website where users can post questions on a wide range of subjects, and community members provide the answers. The person who started the thread, and other community members, can vote on the various answers provided for each question. Users can then achieve higher rankings based on how many useful answers they have provided. Also the more karma points a user has, the more trust worthy his or her answers may appear. (Harper et al., 2007; StackOverflow Badges, 2010; StackOverflow FAQ, 2010).

StackOverfow is a question and answer site where a person can ask a technical question and get an answer from a huge community of technical experts. Each person in the community can vote on an answer. The answer that gets the most votes gets selected as the preferred answer. Users earn reputation points and karma badges based on how many votes their answers get. The karma badges and reputation points of a user are displayed in his/her profile page, along with his/her profile picture. A graph of the user's activity and reputation is also provided. In this technical site, reputation and badges are a big draw and motivator. By making their profile information public, the users make a name for themselves as well as prove their expertise over other users in the community.

Motivation is important for FOSS projects, as pointed out by Raymond in the Cathedral and the Bazaar. A FOSS project is never finished, only abandoned when developers lose interest. The "release early and often" mantra proposed by Raymond helps engage the community during the software-testing phase. End-users help to find bugs, test early and provide feedback on the project, thus helping developers and motivating them. FOSS projects that have motivated developers and community members can sometimes see new bug fix updates/releases within the same day or week (Luther & Bruckman, 2008).

A study conducted by Burke and Kraut at Carnegie Mellon University in 2008, found that politeness could triple the reply count on technical messages. It was also noted that perceived politeness varied from community to community. Thus language greatly affected the motivation that developers and community members had when looking at bugs and requests within the various communication channels. A good set of communication rules, and polite language, greatly enhanced community contributions and communication (Burke & Kraut, 2008).

One of the most basic problems that every online community must solve is getting users to participate. Without participation, communities do not exist for very long. Sending invitations to users to encourage them to participate in discussions pertinent to their preferences was found to increase user participation (Harper et al., 2007). Many participants of online communities first started by lurking, and later became more involved. Motivating new participants to become more involved was important for the life of the online community. Many online websites and communities have displayed in user profiles the user's activity as points or karma.

Many websites and online communities provide users with a profile that usually contains a name, email, city, country, user interests, likes, dislikes, hobbies, and answers to other personal questions. A user profile plays a key role in making online connections and friendships. When someone looks at a user profile for the first time, especially if they know very little about that person, the information in the user's profile helps them to decide whether or not this new person might provide an interesting connection. As researchers Dugan and Geyer found out in an experiment conducted by IBM, people who had more interesting and diverse profiles were more likely to have more friends (Dugan et al., 2008).

2.5.7. *Karma systems and FOSS*

Many times in FOSS communities, the reputation of a member depends on what this user is able to give to the community in the form of code, documentation or support (Raymond, 1992). The greater the contribution and time investment that a member puts in, the greater recognition and reputation gained. As pointed out by Dawn Foster in April 29th 2010, the community manager of Meego, a FOSS mobile platform, there are no good metrics with which to quantify or compare contributions of one type to another (Foster, 2010). Some communities may attach more value to coding than documentation; other communities may value forum postings and mailing lists traffic instead of IRC and lurking. The main problem with karma systems is that they are not inclusive of everybody in the community and creating a fair reputation system algorithm has many complexities (Foster, 2010).

The various roles of community members in a FOSS project include: end user, committer, developer, and core developer. These roles resemble karma and reputation systems, but the difference is that there is no visual distinction to highlight the contribution of these members. Only people who are actively monitoring the various channels of communication and code changes can see how a project member moves up through the ranks, based on his/her contributions. The project member then slowly starts to be taken more seriously during discussions and brainstorming meetings (Foster, 2010; Raymond, 1992).

The Apache Software Foundation (ASF) uses a shared leadership model instead of having a strong project leader. In ASF, there are six levels of increasing status and reputation: developer, committer, project manager, committee member, ASF member, and ASF board member (Apache Incubator, 2010). The level that a user has in ASF demonstrates the user's level of commitment to ASF. The higher the rank in ASF the more power a user has. For example, when a user is a committer, he/she doesn't have to go through channels to have code reviewed before it's accepted. Instead, a user with committer status can submit code directly to the repository. In higher ranks, such as ASF Board director, a member must be nominated and then a vote has to take place before the person is given this reputation level. Not surprisingly, this rank carries a high level of prestige and recognition within the ASF community.

As mentioned by Marc Delisie, an active developer of PHPMyAdmin, a PHP frontend to the popular FOSS mysql database server:

> *"Sometimes, coding and maintaining FLOSS is an ungrateful 'job'; I must say that seeing my project's name (and my own name) on Ohloh gives me an energy boost to continue the journey"* (Ohloh, 2010).

Ohloh (http://www.ohloh.net/) is a community, a directory, and a wiki-like site that documents statistics on both FOSS projects and users' activity. A user can create an account in Ohloh, create a new project and point that Ohloh project to the FOSS SVN repository. Ohloh's backend will data-mine and store information about the various activities of the code repository. Ohloh displays this information in users' and projects' profiles and calculates karma points for each user based on how active he/she is in projects. Users can compare both themselves and projects, and see how they stand in ranking.

In contrast to Ohloh, Launchpad (https://launchpad.net/) is a collaboration tool that provides bug tracking, code-hosting, code reviews, ubuntu package hosting and building, translations, mailing lists, answer tracking, and FAQs and Specification Tracking. One of the features that make Launchpad unique is that it brings FOSS projects together, regardless of the tools they use. Launchpad can hook into a FOSS project's repository and bug tracker, allowing developers and end users to work together on inter-related bugs, thus helping the community track bugs that affect various projects. Launchpad also calculates karma points and awards badges to users by taking into account user actions, such as code commits, bug tracker activity, translation, and providing support. Launchpad hooks into the tools of a FOSS project and can thus provide a more accurate karma calculation. One of the interesting approaches that Launchpad has brought to reputation systems is that karma points decay over time. If you are not active in the community, you start losing points, thus allowing new members to catch up with long-time members. This spin on karma provides a more accurate representation of a user's current activity level in the FOSS community (Launchpad Karma, 2010; Launchpad Mailing Lists, 2010).

In IRC, people can get higher privileges, or a special cloak (a flag on their username), to show they are not just any regular user. This allows them to manage a channel, set a topic for a channel, ban users, and prevent users from talking to or kicking users from an IRC channel. These special roles are displayed next to the username in the channel, and is a visual symbol recognized throughout many channels (Teaching Open Source, 2010).

It is common to see FOSS communities keep track of karma or reputation points, but focus on only one aspect of participation, such as forum activity, rating comments or code commits. For example, many FOSS forums, such as Ubuntu and Gentoo, keep track of the number of posts and their quality. This is usually provided by the forum software used (MyBB Feature Tour, 2010). Other forum software allows end users to give each other karma if the user has provided helpful comments or great insight. Launchpad is the only FOSS site that integrates multiple types of contribution when calculating karma points. Launchpad takes into account forum comments, bug tracker activity, code commits and more (Launchpad Karma, 2010).

2.6. Karma Systems and FOSS Newbies

The health and long-term survival of FOSS projects depends on their ability to motivate and attract new developers (Raymond, 1992). People who are interested in self-development are motivated by the learning opportunity and challenges presented by FOSS projects and this helps keep them engaged and wanting to contribute more (Hutson, 2008). Unfortunately not everyone is motivated by self-development.

One strategy for attracting new users and motivating existing ones is to identify easy-to-fix bugs or to document problems to attract new developers. Many FOSS projects do this, which helps direct new developers' attention to introductory tasks and thus build recognition in the community by performing several easy tasks (Google Summer of Code, 2010). FOSS projects such as Linux Kernel Newbies focuses on helping programmers, who are interested in kernel development, learn the ropes. The Linux Kernel Janitors project focuses on going through the kernel source and doing clean ups, code reviews, fixing unmaintained code and various other tasks. Beginning programmers, who start by doing these low level tasks, can receive recognition in the kernel mailing lists and get noticed by the amount of work and contributions they provide (Kernel Janitors, 2010; Kernel Newbies, 2010).

2.7. Karma systems as a means of contribution moderation

Big online communities and FOSS projects get a lot of traffic and contributions in the form of comments to postings, new bugs, and wiki page submissions. These often have a problem with scaling content moderation. The task of filtering and flagging invalid or inappropriate content becomes too much to handle for core members (Lampe & Resnick, 2004). Communities have tried fixing this problem by using automatic methods that look at post frequency, length of post, number of links and keywords in the post. These methods are helpful, but have shortcomings in that they aren't able to catch all trolls, a person who posts inflammatory or extraneous information to disrupt conversations, or off-topic discussions. Online communities often have moderators who monitor content to ensure users abide by community standards.

A study done by Lampe and Resnick, in 2004 at the University of Michigan found that in big online forums, such as Slashdot, some posts got little or no attention from moderators, and moderation mistakes would take a long time to be reversed, or not be reversed at all. In order to mitigate some of these problems, Slashdot created a moderation system based on karma points. Each post on Slashdot can have a rating from -1 to +5 and the default score of a post is set to +1. Posts from anonymous users have a rating of 0. Slashdot users earn karma by participating in the community through moderating comments, reading comments, and posting comments that get a high or low score. Posts from users with high karma start with a rating of +2. Users with low karma have posts starting at 0 or -1. Moderators can then browse the list of posts and either increase or decrease the rating of a post. Slashdot users can achieve the title of moderator by having a high karma score. Even with a karma-based moderation system, Slashdot still relies on a paid staff to act as meta-moderators to ensure that moderators are giving fair ratings to users' posts.

Lampe and Resnick observed certain problems with Slashdot's karma-based moderation system. Moderators were influenced by previous ratings by other moderators, and they were more likely to stay silent if they disagreed with the review. Posts made by users who had no karma because they were anonymous were more likely to end up with low scores. Postings made towards the end of a discussion were more likely to get less attention and it took longer to find hidden gems in the comments. Even though the karma based moderation system had its flaws, it was necessary due to the scale of the site. Meta-moderators agreed with moderators 92% of the time (Lampe & Resnick, 2004).

In Wikipedia and other sites that depend on community contributions, users can have different roles, such as readers, editors, administrators, recent changes patrollers, policy makers, subject area experts, content maintainers, software developers, system operators and more. Most people begin with the role of editor or uploader and after they have mastered that role, they can apply to other roles. The path to achieving administrator status depends on the wiki, but in the English Wikipedia it goes up for a vote. Anybody can vote and if the approval is 80% or more the person gets accepted, between 75%-80% it is up to the bureaucrat. If the acceptance level is less than 75% the person is rejected. People's expectations of users in an administrative role is high, such as having a minimum of 3,000 edits, being with the community longer than three months, and not having received any disputes regarding their contributions (Riehle, 2006).

The roles in Wikipedia come with responsibility and tasks that need to be taken care of. Sometimes the roles are not enough motivation and, as in the case of German Wikipedia, some authors try to become featured authors by getting their work posted on the homepage, thus receiving more recognition from the community. The German Wikipedia struggles with quality assurance, and to address these problems, users try to delete new articles that are not relevant or that don't meet quality standards. To motivate existing editors and content submissions, the German Wikipedia recognizes people who make good contributions with an 'excellent articles' label. They also attempt to motivate other editors by highlighting their work in a section called 'collaboration of the week.' The German Wikipedia has talked about implementing reputation systems to help control the quality of articles and edits, but nothing has been implemented, even though people have shown an interest in this.

With the increasing amount of user-generated content on sites such as YouTube and Wikipedia, the problem of preventing vandalism, spam and providing incentives for quality content increases (Chatterjee et al., 2008). A separate approach to moderating large amounts of user-contributed content in Wikipedia was explored by Adler and Alfaro, who suggested a content-driven reputation system. The basic concept of their system was that users who had contributions that survived a long time, gained reputation. On the other hand, users with short-lived contributions lost reputation (Adler & Alfaro, 2007). Chatterjee and Alfaro further studied some of the issues with content-driven reputation systems. One of the vulnerabilities of these systems was to sybil attacks, where a user with multiple accounts on the system coordinated actions to increase his/her karma points (Chatterjee et al., 2008). The authors argued that content edits that required no effort such as additions or deletions of arbitrary text, were not useful. Also splitting contributions into multiple complex edits should not increase a person's karma any more than if the article had been edited just once.

2.7.1. Problems with Karma Systems as a means of moderation

Another online community that has a huge flux of content are peer-to-peer (P2P) file sharing sites. Studies have shown that many of the files and content shared in P2P networks is mislabeled or corrupted, thus wasting bandwidth (Walsh & Sirer, 2005). Some P2P networks use karma-based systems to mitigate this problem and help users decide whether or not to trust the content they are about to download. Unfortunately, karma alone can not help distinguish between good or bad content. One approach used to solve this problem, as described by Walsh and Sirer, was to use votes, in addition to karma-based systems. Karma systems occasionally get abused by users originally posting good content, but who later post mislabeled or corrupted content.

Kerschbaum published a paper that explored issues with users trying to suppress their low ratings. The author focused on protecting how a user was rated versus who rated the user by using a private feedback system where the ratee and the reputation system did not know the rating. This helped foster more honest and coercion-free ratings. To prevent users from attacking or misusing the system (leaving bad feedback on purpose to harm a person), people could only leave feedback after a transaction and verifying their identity. The approach suggested by Kerschbaum was to not relate a user rating to any one person. In other karma systems a person receiving a bad rating could threaten the rater with negative ratings in return. In the system proposed by Kerschbaum, ratings were public, and only updated after a new batch of ratings were provided. This prevented the rated from checking their ratings before and after a transaction to learn how a transaction affected their rating (Kerschbaum, 2009).

2.8. Reputation Systems outside of FOSS

Even though, there hasn't been much published research on reputation/karma systems in FOSS, these systems have been used for decades outside of FOSS and have become popular in online communities. As discussed before, websites such as Yahoo Answers and StackOverflow allow people to post questions, and community members post answers. People earn badges by answering question correctly or by having their answers marked as useful. These sites keep a leaderboard and display the badges that users earn (StackOverflow FAQ, 2010; StackOverflow Badges, 2010). The Boy Scouts of America started the merit badge program in 1910. In this program scouts learn about crafts, trades, sports, science and future careers and earn related badges along the way. There are more than 100 merit badges to be earned, and each badge has a set list of requirements that need to be fulfilled before the young scout can earn the badge (Boy Scouts of America – Introduction to Merit Badges, 2010).

Foursquare and HomeRun are two companies harnessing the power of mobile devices' GPS, social networking and badges to motivate people to explore their neighborhoods and local businesses. Each time a person visits a participating business, the user checks in using an application in his/her mobile phone to earn points towards badges that can translate into rewards. On Foursquare, the person with the most points earns the badge of "mayor" for that business and gets discounts by showing his/her badge to the store clerks. Once a user has earned enough points or has checked-in frequently, they are eligible for higher discounts and savings. On HomeRun, people share daily tips and savings applicable to participating businesses. These systems, combined with social networking, motivate people to collect and compete for badges with their friends while getting extra savings and discounts (McCarthy, 2010; FourSquare, 2010; HomeRun, 2010).

The Huffington Post has a badge system to motivate and encourage members to share their online activities with friends by connecting their Huffington post account to other social networks. The networker badge is given to users after they connect their Huffington Post accounts to either Facebook or Twitter. The superuser badge is given to readers after they have commented on stories and shared the stories or commented on others via social networks. The moderator badge is achieved once readers flag at least 20 comments online.

2.9. Disadvantages of Reputation Systems

Hoffman studied the various attack and defense techniques against online reputation systems (Hoffman et al., 2009). He observed five different classes of attacks: self-promoting, white-washing, slandering, orchestrated, and denial of service.

Self-promoting involve attackers who change their karma by falsely increasing it. This technique exploits reputation systems that take into account positive feedback (Hoffman et al., 2009). The target of the attacker is the karma formula, but they also exploit other weaknesses in the system related to the calculation of karma points. Similarly, sybil attacks, where a user with multiple accounts tries to coordinate actions to increase his or her karma points (Chatterjee et al., 2008). This increase in karma points can later be used to perform activities that benefit the user at the expense of others in the community. Members of online communities also carry out self-promoting attacks by teaming up and providing dishonest reports for each other to increase their karma (Resnick et al., 2000). Self-promoting attacks can be stopped by only allowing feedback after successful transactions. Such a reputation system prevents people from rating each other too often if they only interact with a limited set of the community (Chatterjee et al., 2008).

The second type of attack is white-washing, where a user exploits a weakness in the system to repair his/her reputation. Once the attacker's reputation is repaired, he/she returns to doing malicious behaviors. Friedman and Resnick, used game-theory analysis to find the limitations of reputation systems when people were allowed to start their reputation over through the use of a new identity. People with no feedback or little history online are typically not trusted as much by peers. One solution to this problem is not allowing people to change their identities, use real names or use once in a lifetime online pseudonyms (Friedman et al., 2000). Karma systems that focus only on negative feedback are vulnerable to this type of attack because people who are relatively new to the community have similar reputations to people with positive long-term behavior.

With slander type attacks, one or more community members falsely provide negative feedback about other members of a community. Reputation systems that don't verify the origin of the feedback are particularly vulnerable to this type of attack. Fear of retaliation through negative feedback prevents users from leaving negative feedback. People with unsatisfactory interactions online have a strong desire to keep things simple and try to prevent future negative and confrontational interactions, thus keeping negative online experiences quiet. As discussed before, this issue is mitigated by updating reputation points in bulk after five to ten new pieces of feedback have been received (Chatterjee et al., 2008), (Hoffman et al., 2009). Depending on the type of community and the monetary value of a good reputation, slandering attacks can be very harmful toward a company's ability to sell its product and earn a customer's trust (Pavlou et al., 2005). A defense against slandering attacks is to verify that feedback is tied to a certified customer transaction and this prevents users from assuming multiple online pseudonyms (Friedman et al., 2000).

Orchestrated, is a type of attack where attackers orchestrate their efforts to combine several of the previous attack strategies (Hoffman et al., 2009). Different members in an online community take turns performing a type of attack and when their karma gets too low, they attempt to repair it. Once the attacker's karma is repaired, they switch roles and perform a different attack strategy. The orchestrated types of attack are harder to mitigate because community members could be providing negative karma towards a party they are not related to at all. To defend against orchestrated attacks, strong reputation systems need to be in place to prevent groups from repairing their karma. Also, the analysis of karma feedback needs to be more thorough, to reveal relationship between parties that could be working together.

The last type of attack identified by Hoffman, is denial of service. In denial of service attacks, people focus their efforts on preventing the karma system from disseminating or updating karma points. Reputation systems who rely on a centralized server to distribute karma points, or information regarding users are the most vulnerable to this type of attack. The best way to mitigate this issue was by providing redundant methods to deliver karma information, or by having a distributed delivery network where trusted users can deliver karma information to the online community (Hoffman et al., 2009).

A different type of issue with reputation systems is lack of motivation. When people left feedback online after a transaction, there is very little incentive to spend the time and leave feedback regarding a seller. People who have a negative interaction with a seller or buyer online are more motivated to fill out the feedback form and leave information behind, regarding a transaction. Thus feedback online can often be biased toward negativity (Resnick et al., 2000). Along with this drawback of karma systems, in online websites such as Amazon and eBay, it is common for buyers and sellers to negotiate and mediate bad experiences before leaving bad feedback. As a consequence, it is mostly the really bad interactions and transactions that get reported through feedback forms (Resnick et al., 2000).

Lack of portability of karma points between online communities is also a drawback of most reputation systems. When Amazon.com first started their karma system, they allowed users to import their eBay reputation history, to let users take advantage of the work and effort they had put into building their eBay karma. Shortly after providing this functionality, Amazon.com stopped letting users import their eBay ratings, due to complaints from eBay that claimed their reputation system was proprietary information.

Finally, displaying and visualizing the karma information to help users make informed decisions is a task that is still hard to mitigate to this day. Most karma systems online only let people provide a rating with preset values, such as 1-5, poor-excellent. Yet these ratings fail to provide information about the person providing the feedback, such as what their karma score is, how long they have been part of the community, or whether or not the feedback came from a bad or relative simple transaction.

3. FOSS at OSU

Oregon State University is one of the schools that provide the best FOSS related curriculum. Students are exposed to FOSS tools in classes, or can gain experience with FOSS at the Open Source Lab (OSL) or help spread the adoption of FOSS by working with the Open Source Education Lab (OSEL). This chapter focuses on the current state of FOSS and history of Beaversource at OSU.

3.1. Collaboration between Departments

With cuts in budget throughout Oregon State University, trying to do more with less is the norm. In 2008, the College of EECS (Electrical Engineering and Computer Science), UHDS (University Housing and Dining Services), Network Engineering Team (NET), and CWS (Central Web Services) at OSU were working on projects, which could benefit from having a central place to share code and collaborate. The people writing code needed a way to easily share their ideas and contributions with each other. EECS was interested in having an area where students could work on class, research and recreational projects, while gaining experience on how to use project management tools, such as bug trackers, code repositories, wikis, and forums.

There have always been many exciting groups and projects going on at OSU, but students don't always know what's available or where to start to look for interesting projects or groups to join. Information was usually spread by word of mouth, or through groups that recruited from freshman classes. The closest thing to a catalog was the list of student groups registered with Student Leadership and Involvement (http://oregonstate.edu/sli/student-organizations/find-organization-you). Research groups didn’t have to register through an entity at OSU, and students discovered them through word of mouth, news stories or student group fairs. Except for the various OSU communities in Facebook and other social networks, people couldn't find peers with similar interests at OSU outside of classes.

Many projects required expertise or help from different disciplines and departments, but it was hard for staff and faculty to connect with students looking for job/research opportunities. The closest thing was to search for 'jobs' (http://search.oregonstate.edu/index.php?q=jobs&site=All), but there were multiple sites at OSU posting jobs and there was no central hub.

3.2. Code/Projects in Beaversource

Our goal was to create an online tool where staff, faculty and students, within OSU, could collaborate on projects together by sharing code, ideas and resources, while using real-world tools like SVN, wiki, bug trackers, and project history. Many staff, faculty and students within OSU had separate instances of these tools spread around that weren't always up to date with security patches and, being isolated, they couldn't share their work. Departments within OSU would develop programs, such as Single Sign On (SSO, http://oregonstate.edu/cws/single-sign-sso-library), that allowed people to authenticate using their ONID account, but if staff didn't share their modifications of these programs, other staff would end up reinventing the wheel. When a new version of a program or library was updated, there was no way for developers to notify other programmers on campus using the program to download the new version.

In the spring of 2007, a couple of graduate students from EECS started to work on sketches and prototypes of this centralized project repository and Beaversource was born. The name Beaversource was chosen because it didn't use a specific technology in the name and the component 'beaver' branded it as an OSU project. The 'source' component had to do with the code-hosting aspect.

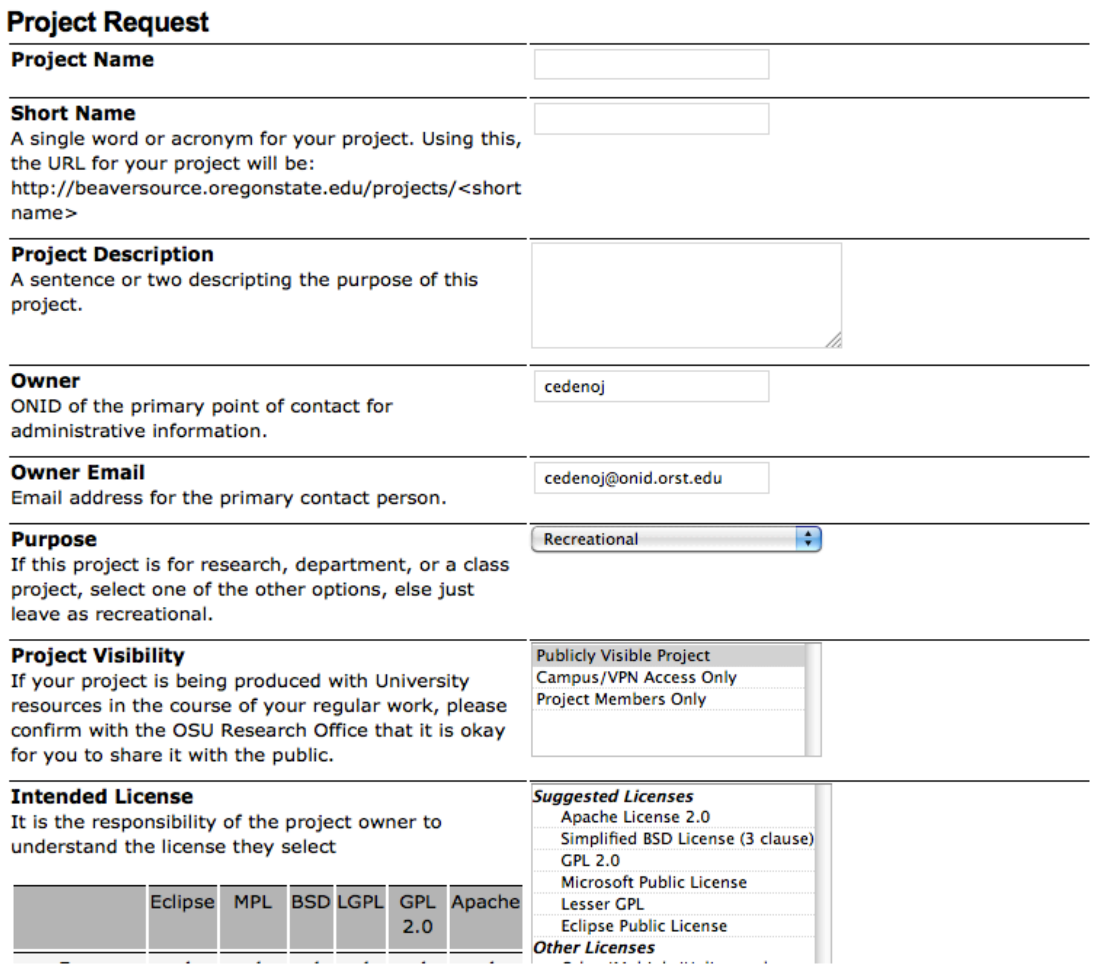
We chose to use Trac (http://trac.edgewall.org/) as the foundation of Beaversource because it integrated all the code-hosting tools needed in one package (see Table 2.1). Trac's functionality could be extended by downloading plugins from the web (http://trac-hacks.org/) or by writing new ones from scratch. The various features listed in Table 2.1 could easily be referenced from each other using shortcuts, and wiki-style editing was part of the bug tracker. This was a superior solution to running separate programs that wouldn't have integrated well with each other.

Table 3.1 Trac features

Needs	**Trac Provides**
Wiki	TracWiki
Bug Tracker	Ticket System
Code Repository	Can hook into svn or git to display repository
Project Statistics	Several plugins display statistics
Others	Project Roadmap Search Reporting

There was a small problem to using Trac. It only supported one project per installation. This wouldn't scale since it was possible that hundreds of projects would be added to Beaversource and this would eventually create a maintenance problem. Fortunately, there is a hack/work around to this problem (http://trac.edgewall.org/wiki/TracMultipleProjects/MultipleEnvironmentsSingleDatabase). Using PostgreSQL (http://www.postgresql.org/) as the backend database and sql views, Trac could be tricked into thinking that there were multiple installations, when in reality there was only one.

Various people from CWS, UHDS, NET, ONID and EECS worked together to get Trac installed. In a matter of a few weeks, an early alpha version was ready for testing. A simple page, listing the projects' names, descriptions, urls, and members was created as well as an online management tool to process and accept project requests. People had to enter their ONID usernames and passwords to gain access to the request form (http://beaversource.oregonstate.edu/request/). Figure 3.1 is a screenshot of the request form where users could enter a project name, short name, description, owner, project members, purpose and license.



Project Request

Project Name	
Short Name A single word or acronym for your project. Using this, the URL for your project will be: http://beaversource.oregonstate.edu/projects/<short name>	
Project Description A sentence or two descripting the purpose of this project.	
Owner ONID of the primary point of contact for administrative information.	cedenoj
Owner Email Email address for the primary contact person.	cedenoj@onid.orst.edu
Purpose If this project is for research, department, or a class project, select one of the other options, else just leave as recreational.	Recreational
Project Visibility If your project is being produced with University resources in the course of your regular work, please confirm with the OSU Research Office that it is okay for you to share it with the public.	Publicly Visible Project Campus/VPN Access Only Project Members Only
Intended License It is the responsibility of the project owner to understand the license they select	*Suggested Licenses* Apache License 2.0 Simplified BSD License (3 clause) GPL 2.0 Microsoft Public License Lesser GPL Eclipse Public License *Other Licenses*

	Eclipse	MPL	BSD	LGPL	GPL 2.0	Apache

Figure 3.1 New Project Request Form

3.3. Social in Beaversource

We then decided to add social networking features to make it easier to find projects, groups and people. By using social networking we wanted to make it easier for people to keep track of groups and the projects they were working on. The social side was meant to augment the code-hosting tools. While many sites like Google and

Sourceforge provided code-hosting, Beaversource is unique with its mix of social networking and code-hosting.

Social networking got right to the heart of Beaversource's goal of helping users find peers with the same interests. It also helped people keep track of the progress friends were making on their projects. As students work on different projects throughout their education, their profile pages became like a portfolio, showing the projects and groups they have worked on. Future employers can look at students' projects to see the code students have developed and to assess the students' skills.

The social side also allows teachers to use Beaversource for some of their classes. Teaching Assistants (TAs) use Beaversource to hold help sessions, answer questions and communicate with students. Classes discuss homework, tests or other class materials using forums. Teachers use the File upload functionality to add their PowerPoint presentations online and they could also create regular webpages.

The social networking side was implemented using Elgg (http://elgg.org/). At the time this was the only viable solution. Elgg is a social networking framework written in PHP that provided the basic features needed (see Table 2.2) and could be extended through the use of plugins. Alpha versions of Beaversource were based on elgg 0.8.x

Table 3.2 Elgg Features

Elgg Features	**Helps toward goals**
Blogs	allow users to post in their blogs to make announcements for their projects and share information with peers
User profiles	allow users to have an identity in the site with name, contact information, description and pictures
User Listings	facilitates the browsing of peers and people to collaborate in projects.
Groups	helps users with similar interests create a place to gather.
Group Listings	makes it easier to browse through projects

Forums	Teachers and students could have online discussions.
Friends	aids users keep track of peers with similar interests or friends from classes.

4. Methodology Overview

To learn more about the needs of Beaversource users, a survey was developed. This was the first time the Beaversource community had been sent a survey. This survey was used to gather demographic information as well as learn more about the motivation behind the use of the community site and its tools. This chapter will start with a description and discussion of the research questions. Next an overview of the survey will be presented

4.1. Research Questions

To tend to the goals this research project, as defined in the introduction chapter, making it easier to identify good peers to work with; promoting exemplary behaviors; balancing the motivating efforts across a variety of activities; and nudging users to explore the site, three research questions were formulated.

4.1.1. RQ1: Can a karma system boost activity in the site?

With more than 1,200 users, Beaversource is capable of providing services to a lot of students in the OSU community. Yet, site statistics show that many users don't use the site very frequently. I want to find out whether or not using game mechanics in the form of a karma system can increase the user activity within the site. Is it possible to get people excited about their karma badges and get them to want to work hard so they can earn the next karma level?

4.1.2. RQ2: Is it possible to reward social activity through karma badges?

Beaversource has two components: code hosting and social networking. The code hosting consists of code commits, bugs and wiki entries, which means it's quantitative by nature and easier to measure and reward. The social networking side deals more with friendship connections and interacting with other people in the site, which makes it qualitative and harder to measure. In the literature review, it was shown that a couple of companies (Xbox and Foursquare) are successfully using badge systems to motivate users. I want to know if a similar approach can work in Beaversource. Motivating people to use the social networking could be hard, but the benefits can enhance the user experience.

4.1.3. RQ3: Can a karma system reward multiple types of activity fairly?

The literature review shed light on a weak area of most karma implementations in FOSS communities. Most reputation systems focus on just one type of activities and try to reward these activities while ignoring other categories. Beaversource is a diverse community where some people write code, interact with friends, work on documentation or use it as a means of communication. I think achieving a balance that motivates different categories of behaviors is crucial to not make users feel excluded. Communities and projects value different types of contributions very differently, which will make rewarding multiple types of activity fairly a challenge.

4.2. Survey

The first step to help us answer the research questions was to develop a survey. The overarching goal of the survey was to establish a baseline understanding of community member's needs and to gauge interest in some of the features of the site, such as social networking and code-hosting. The intent here was to see what areas of Beaversource could or should be improved while using our limited resources efficiently. With this being the first survey of the community, the questionnaire was kept broad in order to gather as much information as possible. To motivate student participation, users provided their email at the end of the survey to enter into a raffle for a chance to win one of three $20 OSU Bookstore gift certificates. The online survey was divided into three sections six demographics questions; five computer experience and background questions and 15 questions regarding Beaversource use. See Table 3.1, below for the full list of questions.

Table 4.1 List of survey questions

Demographics	
Question	**Answers**
What is your gender?	Select **one** of the following Male Female
What is your age?	Select **one** of the following 18-24 25-34 35-44 45-54 55+
What is your status at OSU?	Select **one** of the following Undergraduate student Graduate student Post Doc Faculty/Staff Alumni Other
If you're an undergraduate student, what year are you in?	Select **one** of the following Freshman Sophomore Junior Senior
If you're a student, please select your major. If you're a staff or faculty, please select your department.	Select **one** of the following Long list of majors and departments
How many computer science classes have you taken in college?	Select **one** of the following 0 1-3 4-6 7-9 10+
Computer Use	
Question	**Answers**
Approximately how many hours a day do you spend on a computer?	Select **one** of the following Less than 1 hour 1-2 hours 3-4 hours 5-6 hours 7-8 hours More than 8 hours

Do you use any of the following social networking sites?	<u>Please choose **all** that apply</u> MySpace Facebook Ning Windows Live Spaces LinkedIn Twitter Flickr Last.fm Beaversource Orkut I use no social networking sites Other
How often do you use social networking sites (excluding Beaversource)?	<u>Select **one** of the following</u> Never Rarely Occasionally Several times a week Several times a day
Beaversource Experience	
Do you use any of the following source code-hosting websites?	<u>Please choose **all** that apply</u> Google Code SourceForge CodePlex GitHub Freshmeat Apache Incubator Beaversource I use no source code-hosting sites Other
How often do you use source code-hosting sites (excluding Beaversource)?	<u>Select **one** of the following</u> Never Rarely Occasionally Several times a week Several times a day
How long have you been a member of Beaversource?	<u>Select **one** of the following</u> I'm not a member Less than 1 term 1-2 terms 3 terms – 1 year More than 1 year

How did you learn about Beaversource?	<u>Select **one** of the following</u> Class Email announcement Word of Mouth Barcamp From a search engine or another website Other
How often do you use Beaversource?	<u>Select **one** of the following</u> Never Rarely Occasionally Several times a week Several times a day Other
Are you member of groups or projects in Beaversource?	<u>Select **one** of the following</u> Member of a group Member of a project Member of both a group and a project Neither member of a project or group
How many projects do you currently belong to on Beaversource?	<u>Select **one** of the following</u> 0 1-3 4-6 7-9 10+
How many groups do you currently belong to on Beaversource?	<u>Select **one** of the following</u> 0 1-3 4-6 7-9 10+
How many friends do you currently have on Beaversource?	<u>Select **one** of the following</u> 0 1-5 6-10 11-15 16-20 21+
How often do you interact (contact friends, post on their wall, or participate in forums) with your friends or groups on Beaversource?	<u>Select **one** of the following</u> Never Rarely Occasionally Several times a week

	Several times a day	
What are your current reasons for using Beaversource?	<u>Please number each box in order of preference from 1 to 5</u> Class Project Social networking Wiki/file hosting Other	
Rate the following features on Beaversource? Please choose the appropriate response for each item:		
Wiki	<u>Usefulness:</u> Strongly agree Agree Neutral Disagree Strongly disagree	<u>Importance:</u> Strongly agree Agree Neutral Disagree Strongly disagree
SVN web viewer for projects	<u>Usefulness:</u> Strongly agree Agree Neutral Disagree Strongly disagree	<u>Importance:</u> Strongly agree Agree Neutral Disagree Strongly disagree
Bug tracker in projects	<u>Usefulness:</u> Strongly agree Agree Neutral Disagree Strongly disagree	<u>Importance:</u> Strongly agree Agree Neutral Disagree Strongly disagree
Project statistics	<u>Usefulness:</u> Strongly agree Agree Neutral Disagree Strongly disagree	<u>Importance:</u> Strongly agree Agree Neutral Disagree Strongly disagree
History (wiki edits, commits, tickets) of projects	<u>Usefulness:</u> Strongly agree Agree Neutral Disagree Strongly disagree	<u>Importance:</u> Strongly agree Agree Neutral Disagree Strongly disagree
Project discussions	<u>Usefulness:</u> Strongly agree Agree	<u>Importance:</u> Strongly agree Agree

	Neutral Disagree Strongly disagree	Neutral Disagree Strongly disagree
Group discussions	Usefulness: Strongly agree Agree Neutral Disagree Strongly disagree	Importance: Strongly agree Agree Neutral Disagree Strongly disagree
User's profiles	Usefulness: Strongly agree Agree Neutral Disagree Strongly disagree	Importance: Strongly agree Agree Neutral Disagree Strongly disagree
User blog	Usefulness: Strongly agree Agree Neutral Disagree Strongly disagree	Importance: Strongly agree Agree Neutral Disagree Strongly disagree
File Uploads	Usefulness: Strongly agree Agree Neutral Disagree Strongly disagree	Importance: Strongly agree Agree Neutral Disagree Strongly disagree
Commit code to a project	Usefulness: Strongly agree Agree Neutral Disagree Strongly disagree	Importance: Strongly agree Agree Neutral Disagree Strongly disagree
Do you feel that the code-hosting tools and social networking features are well integrated?	Select **one** of the following Yes No	
If you selected 'No' in the question above, explain why?		
What would you change in Beaversource?		
What features would you add to Beaversource?		

Which features most urgently need improvement in Beaversource?	<u>Please number each box in order of preference from 1 to 11</u> Wiki SVN web viewer for projects Bug Tracker in projects Project statistics History (wiki edits, commits, tickets) of projects Project discussions Group discussions User's profiles User blog File Uploads Commit code to a project
If you would like to enter our raffle for a chance to win a $20 OSU bookstore gift card, please enter your email.	

Before the survey was sent, it went through several revisions as suggested by the research team. Questions were rephrased; links to more information were provided and dependent questions were clustered to simplify the process for survey respondents (only ask question B if question A's answer is true, etc.). A waiver for signed inform consent was sent to the IRB office at OSU because it was not feasible to collect user's signatures during an online survey. The inform consent was displayed to users before they began the survey, and people had to click the 'I agree' button before they could continue. The college of EECS had an installation of LimeSurvey (http://www.limesurvey.org/), a popular FOSS survey software, which was used to create the survey and code the answers to make analysis easier at the end.

A mailing list was compiled using all the Beaversource community members' email addresses from the Beaversource database. A script was developed to clear out and remove users who were no longer part of the OSU community, most commonly because they had graduated or moved on to other jobs. An email was sent out to the Beaversource community, as well as the school of EECS, to invite users to fill out the survey and to come and celebrate 1,000 users on Beaversource on Jan 14th, 2010 with cake and refreshments, where the survey was also promoted. A link to the survey was additionally posted on the Beaversource homepage. By the end of January 2010, we had collected around 140 survey submissions.

The collected data was cleaned and purged of incomplete data. There were a total of 151 submissions, but 51 of them were incomplete, meaning people didn't go through all the questions and hit submit at the end. The survey software, LimeSurvey, showed blank rows for these incomplete submissions. The survey was designed so respondents did not have to authenticate (enter a username or password) and could not to start the survey, save it and come back later to finish it. This could possibly have lead to the incomplete surveys. In order to assure the user's privacy and anonymity, metadata was not collected regarding respondents who submitted incomplete answers such as their IP address or web browser. A total of 100 valid submissions remained after the data cleaning process was completed. The next step was to analyze the completed survey results.

5. Karma Implementation

In this section, we'll describe the implementation of the reputation system for the Beaversource Community at Oregon State University. We'll start by reviewing the motivation behind it, then we'll analyze the design decisions, next we'll cover the balance criteria for badges, followed by development aspects, and end with procedures.

The motivation behind the reputation system is to reward and promote active community members for their valuable contributions. There are many active community users who use the site to host their school projects or research projects, yet their model behavior goes unrecognized by peers. Helping projects by writing documentation in wiki or submitting tickets doesn't sound as important as providing a bug fix or the addition of new features, but is a key task that needs to be rewarded in all projects.

At the same time, another motivation behind Beaversource's karma system is to help projects identify how desirable a prospective member is. As discussed previously, member's contributions from project to project don't usually follow them. Each time a user goes into a new project they have to build reputation and trust in the community. Gaining the trust and confidence from peers in a project takes time, and our karma system attempts to make this process easier and less time consuming.

The final goal of the karma system is to recognize and motivate people who are active in various areas of the community such as: coding, forums, documentation and social aspects. The majority of users' activity happens in the code-hosting site of Beaversource. By rewarding a variety of behaviors throughout the site, the reputation system wants to encourage users to explore new areas of the site and try

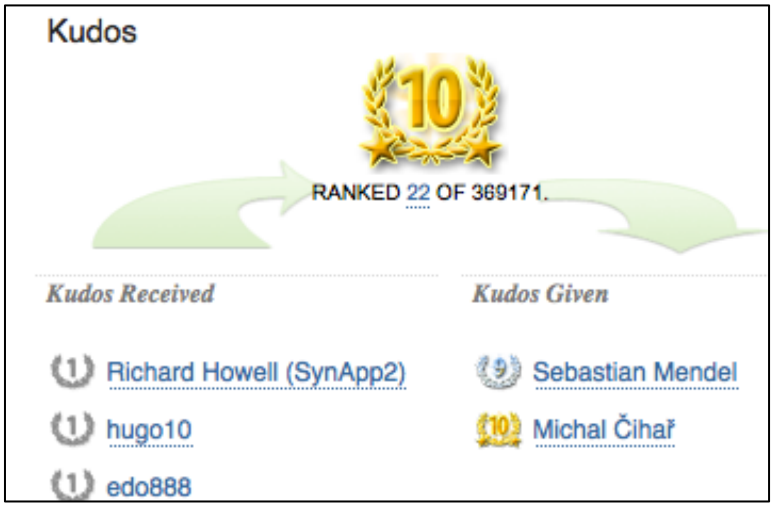


Figure 5.1 Ohloh's karma displayed in users' profile (Ohloh, 2010)

out other functionality that they might not have used otherwise. By encouraging social activity in the community, the karma system wants to make the community site more approachable to non-technical users. Most people think of Beaversource as a place to share code. The high emphasis and activity on technical projects is not welcoming to non EECS students.

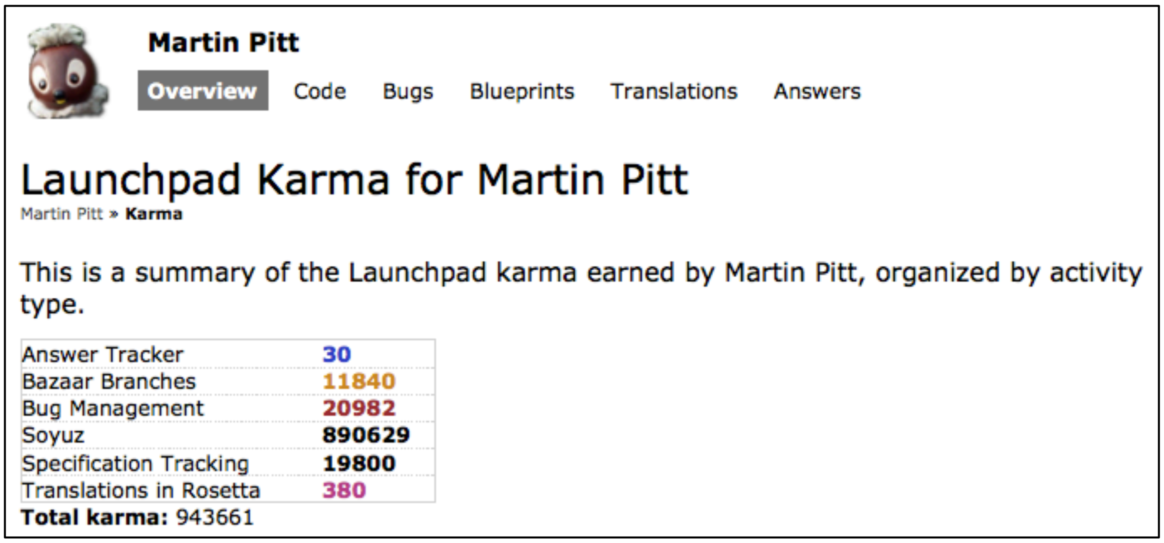
5.1. Design

During the planning and design phase, the karma points formula was kept simple. The badge system was similar to the one provided by Ohloh (http://www.ohloh.net/, see Figure 5.1). Ohloh's reputation system allows users to give kudos to peers in the community, and keeps track of the user rankings in code contributions. Ohloh's karma badge system works in the background and collects data from the source control management system (SVN, CVS or git). In the user profile pages, it's easy to browse and very prominent the various karma badges that an FOSS developer has earned. This approach of performing the data mining and karma calculations in the backend was how the Beaversource karma system's backend scripts were modeled.

Launchpad karma's system (https://launchpad.net/ see Figures 5.2 and 5.3) is proprietary and information about how it works was not available. From visual inspection, we observe that it keeps track of points a user has earned in various categories within the community. It also has badge icons to specify the types of contributions the user made to the communities (see Figure 5.3). Launchpad had the most comprehensive karma system because it looked at a wide variety of behaviors and metrics to award points. We kept this idea under consideration while we designed Beaversource's karma system.

FOSS forum software (e.g. http://www.phpbb.com/ and http://www.mybb.com/) use two factors to calculate karma, the number of posts being the main factor, with labels/badges awarded at key thresholds. The second factor used by FOSS forums is public voting, letting people give/take karma away from other members via a thumbs up/down voting system. Many online communities and projects use FOSS forum software to keep track of discussions. Thus end users have gotten used to the concept of # of posts and labels in forums.

The last example was Elgg's karma system (see Figure 5.4), vazo_karma system (http://elggdev.com/vazco_karma), which uses the same framework as Beaversource. We paid close attention at the design and UI used by vazco_karma. Figure 5.4 is the configuration page for Elgg's vazco_karma plugin, which was the only karma plugin available for Elgg at the time. This plugin was evaluated to see what features it offered and how it handled displaying the karma points and making the list of available karma badges configurable. The first section of Figure 5.4, defines a list of karma badges available. Each line represents a separate badge. The format used to define the badges is: "badge name|required points|icon". The next two settings control whether or not karma badges are displayed in the user profile page and user listing pages. The last section of Figure 5.4 displays the various behaviors in Elgg and has boxes to specify how many points a user earns each time they perform the behavior.


Martin Pitt

Overview Code Bugs Blueprints Translations Answers

Launchpad Karma for Martin Pitt

Martin Pitt » **Karma**

This is a summary of the Launchpad karma earned by Martin Pitt, organized by activity type.

Answer Tracker	30
Bazaar Branches	11840
Bug Management	20982
Soyuz	890629
Specification Tracking	19800
Translations in Rosetta	380

Total karma: 943661

Figure 5.2 Launchpad's Karma page for a user (Launchpad Karma, 2010)

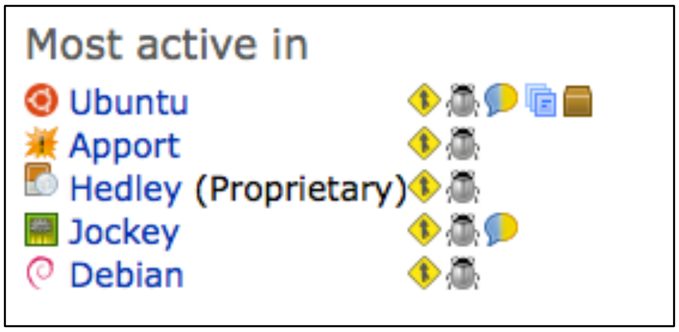


Figure 5.3 Launchpad's identifying lable on user profile (Launchpad Karma, 2010)

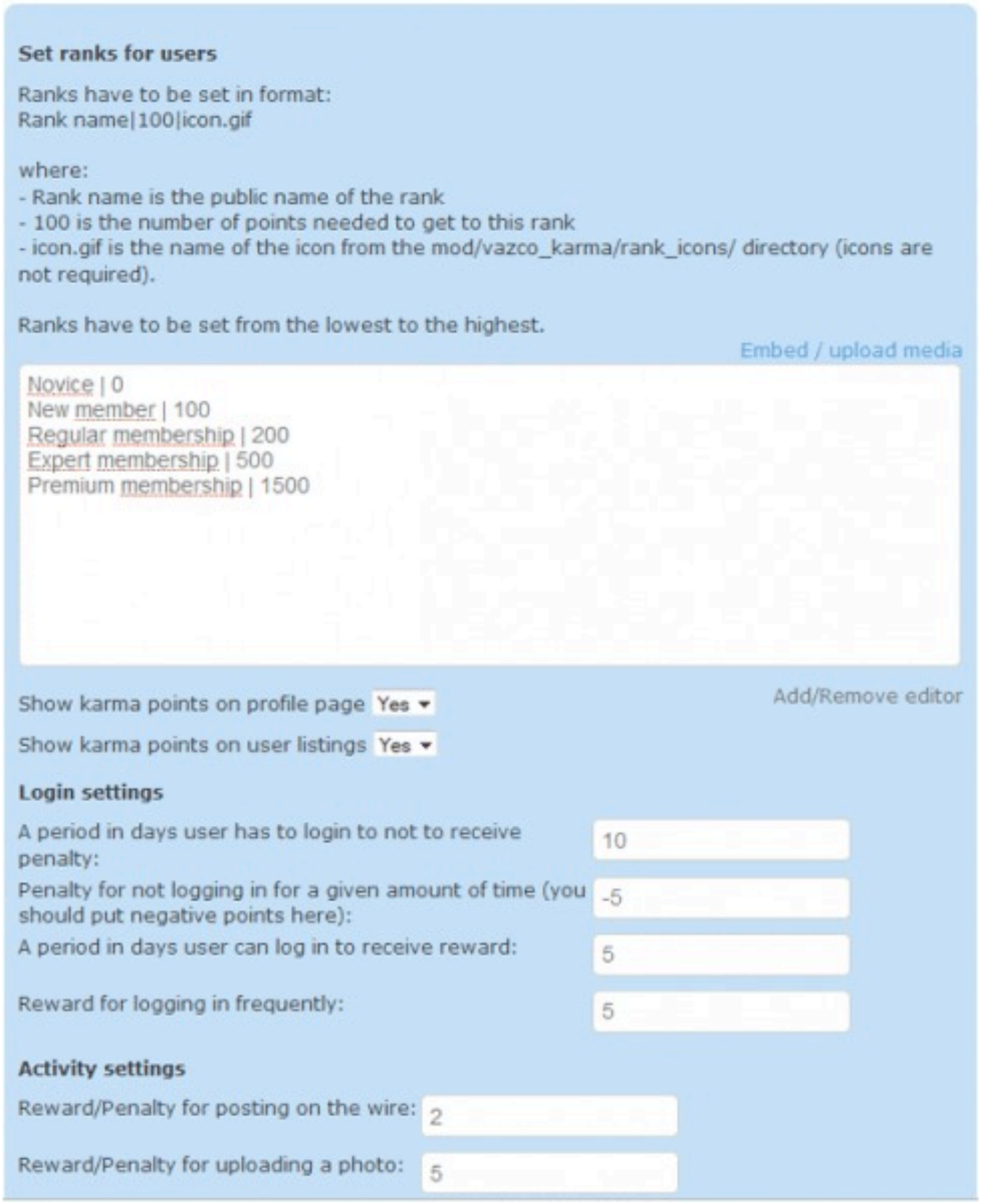
After researching how FOSS communities use karma and reputation systems, behaviors were identified for rewards in Beaversource. A behavior can be classified in two areas: the social side (Elgg) and the code repository side (Trac). See Table 4.1 for the list of actions and how they tie back into the motivations behind the karma system and rewarding the desired behaviors in the community.

Table 5.1 Actions rewarded by Karma system and the motivation behind them

Action	Elgg	Trac	Motivation
Posting on the wire (twitter clone)	X		• Integrate Beaversource with other social networks. • Promote communication between users • Increase the sense of liveliness of the site
Uploading files	X		• Increase use of Beaversource and

			enhance user profiles with activity • Make the community less intimidating to non-tech savvy users
Creating a blog post	X		• Increase the number of postings and user activity displayed in the homepage • Increase communication between users • Increase the sense of liveliness of the site
Posting comments on blogs	X		• Enhance interaction between users in the community • Increase the sense of liveliness of the site
Participating in forums	X		• Boost activity in groups and projects between members • Aid off-topic and casual conversations • Increase the sense of liveliness of the site
Sending messages to users	X		• Raise the frequency of user activity with friends • Improve the friendliness of the site for non-tech savvy users
Receiving messages from users	X		• Raise the frequency of user activity with friends • Improve the friendliness of the site for non-tech savvy users
Someone adding you as a friend	X		• Promote communication with peers from class or projects • Increase the sense of freshness in the site activity
Adding someone as your friend	X		• Promote communication with peers from class or projects • Increase the sense of freshness in the site activity
Creating a group	X		• Multiply the variety of available group forums • Boost available communities for new users to explore • Improve leadership and management skills of users
Joining a group	X		• Expand a person's network of peers • Increase the discussions a user can

			participate on
Creating a project	X		• Provide students with experience on managing projects • Increase students familiarity with code-hosting tools
Joining a project	X		• Promote a wide range of projects to increase students' experience • Increase students familiarity with code-hosting tools
Committing code		X	• Boost activity within projects • Promote smaller atomic commits
Editing a wiki page		X	• Improve user's familiarity with wiki engines • Rewarding a variety of contributions
Creating a ticket		X	• Rewarding a variety of contributions • Provide people with experience on how to write better tickets
Updating a ticket		X	• Rewarding a variety of contributions • Boost activity within projects from non-committers



Set ranks for users

Ranks have to be set in format:
Rank name|100|icon.gif

where:
- Rank name is the public name of the rank
- 100 is the number of points needed to get to this rank
- icon.gif is the name of the icon from the mod/vazco_karma/rank_icons/ directory (icons are not required).

Ranks have to be set from the lowest to the highest.

```
Novice | 0
New member | 100
Regular membership | 200
Expert membership | 500
Premium membership | 1500
```

Show karma points on profile page: Yes

Show karma points on user listings: Yes

Login settings

A period in days user has to login to not to receive penalty: 10

Penalty for not logging in for a given amount of time (you should put negative points here): -5

A period in days user can log in to receive reward: 5

Reward for logging in frequently: 5

Activity settings

Reward/Penalty for posting on the wire: 2

Reward/Penalty for uploading a photo: 5

Figure 5.4 Elgg's Vazco Karma Settings Page

5.1.1. Balancing

Determining the points required for each badge is a subjective decision. The goal of the reputation system is to allow the majority of the members (70%) to have easy access to at least one, but no more than three badges, and the required points for the 1st level badge in each category is low. At the high end of the reputation scale, the levels are harder to achieve and only a small percentage of users (25%) can easily reach those levels. The karma points required significant tweaking as well as trial and error. The required points for each badge were changed frequently and the scripts were run multiple times to figure out the number of people with each karma badge. This continued until the desired number of users for the karma badges was reached. The karma badge system went through three iterations during this thesis project. Throughout the three reputation system iterations, the karma formula was debugged and tested. Some of the bugs caused points not to be calculated correctly for users. These bugs were corrected throughout the iterations and people's karma was adjusted accordingly.

Table 4.2 reports the number of users with karma badges in the system. The total number of karma badges is 1,315. Some users have multiple badges. We can see that the people who have badges, 488 have easy access to the 'Shy' badge and 274 have access to the 'code cowboy'. The 'coding wizard', 'social butterfly' and 'chatter box' badges are a bit hard to achieve.

Table 5.2 Number of users with badges in live site

Category	**Badge Name**	**Number of users**
Friendship	Shy	488
	Charming	154
	Friendly	59
	Total:	**701**
Blogging	Amateur blogger	22
	New and upcoming blogger	4
	Mad blogger	2
	Total:	**28**
Programming	Code cowboy	274
	Code ninja	38
	Coding wizard	11
	Total:	**323**
Tester	Software tester	92

	Total:	**92**
Active Member	Articulate	145
	Chatter Box	18
	Social Butterfly	8
	Total:	**171**

5.2. Development

The code-hosting tools do not have an administrator UI to manage settings, and the settings are on a per project basis. The social site managed by Elgg, provides an administrator UI where users can manage settings for the community as a whole. Elgg's plugin interface allowed us to extend the user listing page and user profile page to display users' karma. Elgg's plugin interface made it easy to store the user's karma as metadata to be retrieved in other parts of the site. For these reasons, the karma management functionality was added to the Elgg framework. Administrators and global settings for Beaversource are managed through either backend scripts or through Elgg settings. Making the karma badge system an Elgg plugin provided a GUI to easily manipulate the social side, which means that it can easily be managed and updated without the aid of a developer.

5.2.1. *First Iteration*

The Beaversource karma plugin was modeled after the vazco_karma plugin mentioned above (Figure 5.4). First, points were assigned to various user actions. The points could be negative (to discourage a behavior), zero or positive (to encourage a behavior). Each time the user performed that behavior those points would be added to their karma. A total number of points would be calculated for a given user, and based on the number of points; a different badge would be given to the user. Only the highest-level badge would be displayed in a user's profile. There were only three karma badges available in this version: Newbie, Amateur and Expert.

This first iteration of the karma system did not allow us to specify what actions contributed towards a given badge. Administrators could specify the required points for a karma badge, and the image to display for the badge (Admin interface was modeled after vazco_karma, see Figure 5.4). Another setting allowed the karma badges to be displayed in user profiles and user listings of the community site. One decision made during the development process was to not make the karma badge list user-customizable. This would ensure that the list of karma badges would be visible for all users in the system in a consistent location and style. The list of karma badges was only displayed on the social side of the website. Trac did not support displaying user icons or extra detail for users, only their username. One problem during the development of the karma badges was the lack of good FOSS icons to choose for the badges. The set of famfam icons was chosen, a popular FOSS icon set used throughout many websites. This first iteration of the karma badge system did not go live, but was available in the development site on Feb 3rd, 2010.

5.2.2. Second Iteration

In the next iteration, the main change was that karma badges could now specify which activities contributed points towards each badge. This change provided greater flexibility for karma badges. Karma badges then only reflected a user's activity in coding, social activity, testing, or group/project activity. Another feature allowed administrators to reward users with a set of points the first time they performed an action. This could be used to create badges, such as 'first commit' and 'first blog post'. Icons used for the karma badges were also revisited, but the icons were still not very good since only the famfam icon set was available. The set of karma badges was also revisited and four categories for badges were identified: blogging, friendship, programming and tester. For more details on the updated list of karma badges see Table 5.3 below. This updated version of the reputation system went live on March 23rd, 2010.

Table 5.3 Categorized set of badges in second iteration

Categories	Badges	Actions
Blogging	Amateur blogger (points req: 1) New and upcoming blogger (points req: 10) Mad blogger (points req: 25)	Posting in blogs (+1)
Friendship	Friendly Beaver (points req: 5) Beaver Buddy (points req: 15) Cozy Beaver (points req: 30)	Adding someone as your friend (+1) Someone adding you as his or her friend (+3) Posting in someone else's message board (+1) Someone posting in your message board (+1)
Programming	Code Cowboy (points req: 25) Code Ninja (points req: 150) Coding Wizard (points req: 300)	Code commits (+1) Project owner (+5) Project member (+5)
Tester	Software Tester (points req: 50)	Creating a ticket (+5) Updating a ticket (+5) Wiki edits (+2)

5.2.3. *Third Iteration*

In the last version of the reputation system, a new category was introduced, Active Member (see Table 5.4 for details), to differentiate between a user being active in the site and having friends. The friendship category was also updated so that users only gained points for having friendship connections. The badges and actions in the Table 5.4 are in addition to the karma categories and badges in iteration two. A page was added to describe the various karma badge categories, and the actions a user needed to take in order to earn the various badges (http://beaversource.oregonstate.edu/social/mod/bsc_karma/pages/karma.php). This version of the karma badge system went live on May 15, 2010. Two weeks later, a script was put in place to keep track of the number of users each badge had.

Table 5.4 Additional karma badges implemented in third iteration

Categories	Badges	Actions
Active Member	Articulate (req. points = 3) Chatter Box (req. points = 15) Social Butterfly (req. points = 30)	Wire posts Posting comments in blogs Forums and sending/receiving messages
Friendship	Friendly Beaver (points req: 5) Beaver Buddy (points req: 15) Cozy Beaver (points req: 30)	Adding someone as your friend (+1) Someone adding you as his or her friend (+3)

5.3. Participants

All users in the system were included in the karma badge system roll out. Users had no choice of whether they wanted to have karma badges displayed on their profile page or not. Adding new features to the community, such as karma badges, is part of the end user agreement. That's why users were not notified or asked about whether they wanted to opt out of the karma badge system. Once the karma system was rolled out, a blog post, a wire post and a message in the homepage was displayed to tell users about the karma system

5.4. Procedures

A script was run every hour to calculate and update the karma for 150 users at a time. Karma calculations were attempted for more users at a time, but time execution limits were encountered when using the Elgg framework. The script that calculated karma took advantage of other backend scripts to gather metadata about social and project activity and calculates the number of times various actions had been performed by a user. The scripts used by the reputation system were written in PHP to query the Trac and Elgg databases as well as to perform the calculations. The karma of a user was stored in the ElggUser object as metadata. This allowed the karma information to be easily displayed in the frontend.

The error handling logic of the backend scripts sent an email to the admin list if the script ran into a problem. After the script finished updating the karma for 150 users, it gathered information on how many users had each karma badge and stored it in a CSV file that was imported into Excel. This allowed researchers to look back and assess how the number of karma badges changed over time, in addition to detecting trends. The karma backend scripts, also exposed an API that allowed project activity information to be gathered for a given project or user. This API, will enable future development of visualizations to aid in the display of statistics such as: active project members today/this week/this month/all time; activity of a project/user over this week or month. One plug-in was developed that allowed for the display of the last 5 commits from all projects.

6. Results

6.1. Survey

As discussed previously, the survey was sent out to Beaversource users to find out how to improve the community site. Out of 1100+ users in the community, 100 completed the survey. This constitutes 10% of the total user population, which means we can provide significant results.

6.1.1. Demographics

All the questions under this section were optional. Table 6.1 provides a summary of the responses.

Table 6.1 Demographics summary

Demographic	**Answer Breakdown**
Gender	Male – 74% Female – 24%
Age	18-24 – 69% 25-34 – 24% 35-44 – 1% 45+ - 6%
Status at OSU	Undergraduate – 70% Graduate – 24% Faculty/Staff – 4% Alumni – 0% Other – 2%
Undergraduates	Freshman – 17% Sophomore – 20% Junior – 23%

	Senior – 40%
Number of CS classes taken in college	0 – 3% 1-3 – 40% 4-6 – 19% 7-9 – 7% 10+ – 31%

Gender

Males make up the majority of the Beaversource community. These results were surprising due to the high percentage of female respondents, which was more than twice the rate of 10% in the EECS female population (http://eecs.oregonstate.edu/). The percentage of female participation in the survey is much higher than the 1.5% in FOSS development. This indicates that the Beaversource community welcomes women's participation.

Age

The age breakdown, in Table 6.1, was not surprising since it matched the student population in EECS. The average age of students is much younger (18-24) than the average age (30) of a FOSS programmer (Gourley, 2009). The assumption was made that the majority of users were familiar with computers, social networking sites and browsing the web.

Status at OSU

The data, in Table 6.1, shows the make-up of survey respondents. Undergraduate and graduate students made up 94% of the survey respondents, staff and faculty accounted for 4%. Alumni do not have access to Beaversource because an OSU account is required for access. 'Others' are non-OSU members who collaborate with OSU faculty, staff or students on projects.

Undergraduate Student Status

The majority of our undergraduate students are Seniors or Juniors, which means they are not new to OSU and already have an established network of friends and resources.

Major/Department Breakdown

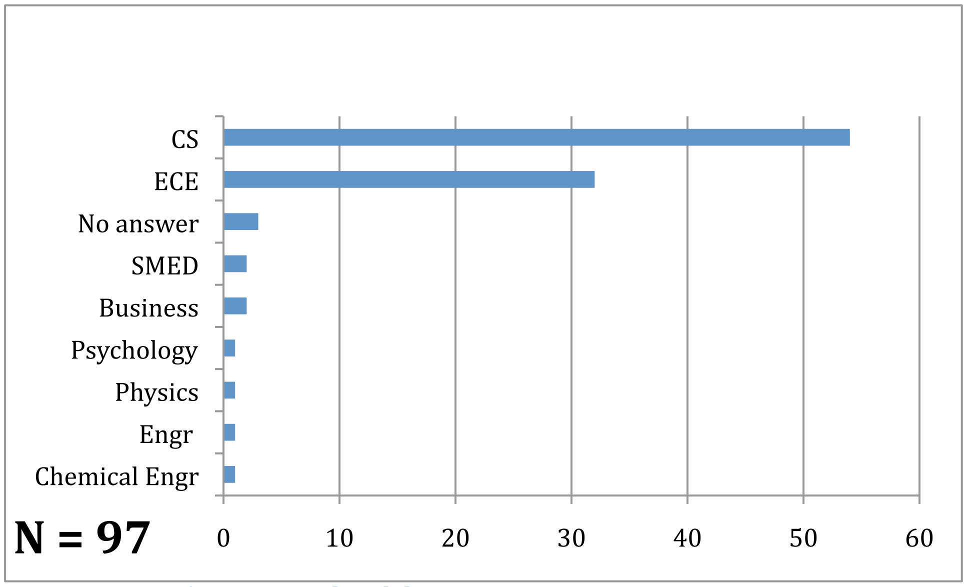


Figure 6.1 Major/Department breakdown

CS and ECE students made up over 80% (78) of the survey respondents. The main draw for them is classes that are using Beaversource. The code-hosting and project features have a lower barrier of entry for students of ECE and CS. We can also assume that people from these disciplines are more interested in the tools provided by Beaversource.

6.2. Computer Use

All the questions in this section were required in the questionnaire.

How many hours a day you spend on a computer daily?

Table 6.2 Hours the user spends on a computer every day

Hours spent on a computer each day	< 1 - 0% 1-2 – 6% 3-4 – 18% 5-6 – 25% 7-8 – 20% 8+ – 31%

With more than fifty percent of the survey respondents spending seven or more hours on a computer per day, and twenty five percent spending five-six hours per day. It was assumed that the user base was familiar to using computers. It was clear that computer usage was an integral part of their lives.

How many friends do you have in Beaversource?

Table 6.3 How many friends people have in Beaversource

How many friends do people have?	**Survey respondents (N = 100)**	**Beaversource site (N = 1250)**
0	32%	70%
1-5	49%	25%
6-10	13%	3%
11-15	4%	1%
16-20	1%	1%

21+	1%	0%

Even though 32% of the respondents have no friends in Beaversource, 49% of them have at least 1-5 friends, and 13% have 6-10 friends. This shows that people clearly have enough friends to be interacting with and making connections.

How often do you interact with friends or groups in Beaversource?

Although the previous question showed that people had friends in Beaversource, Table 6.4 shows that people are rarely using Beaversource to interact with those friends. The last two columns in Table 6.4 show that people who have many friends are more likely to interact with them than people who have projects.

Table 6.4 How often people interact with friends and groups in Beaversource

How often people interact with friends or groups?	Survey respondents	Users with projects	Users with many friends
Never	73%	62%	47%
Rarely	20%	10%	16%
Occasionally	7%	28%	37%
Several times a week	0%	0%	0%
Several times a day	0%	0%	0%

How often do you use social networking and code hosting sites (excluding Beaversource)?

With 65% of the survey respondents using social networking sites several times a week or more frequently, we can assume that the survey respondents are comfortable using social networking features to interact with friends.

In Table 6.5, we see that 13% of survey respondents used code-hosting sites several times a week. This demonstrates the majority of these survey respondents are not taking advantage of wikis, revision control systems, bug trackers, data backup and other features offered by code-hosting sites.

Table 6.5 How often respondents use code hosting and social networking sites

How often do you use? (excluding Beaversource)	Social networking sites	Code-hosting sites
Never	6%	40%
Rareley	10%	50%
Occasionaly	19%	37%
Several times a week	26%	13%
Several times a day	39%	0%

Do you use any of the following networking sites?

People were allowed to pick multiple choices in this question, and that's why the total number of responses was greater than 100. In Figure 6.2, user responses indicated that Facebook was the main competition to Beaversource. The data in Table 6.3, 6.4 and 6.5 backups and explains why not many people are interacting with their friends in Beaversource. Beaversource came to market where there's a strong and well established social networking site and competing against Facebook should be present during the development and design process of new features.

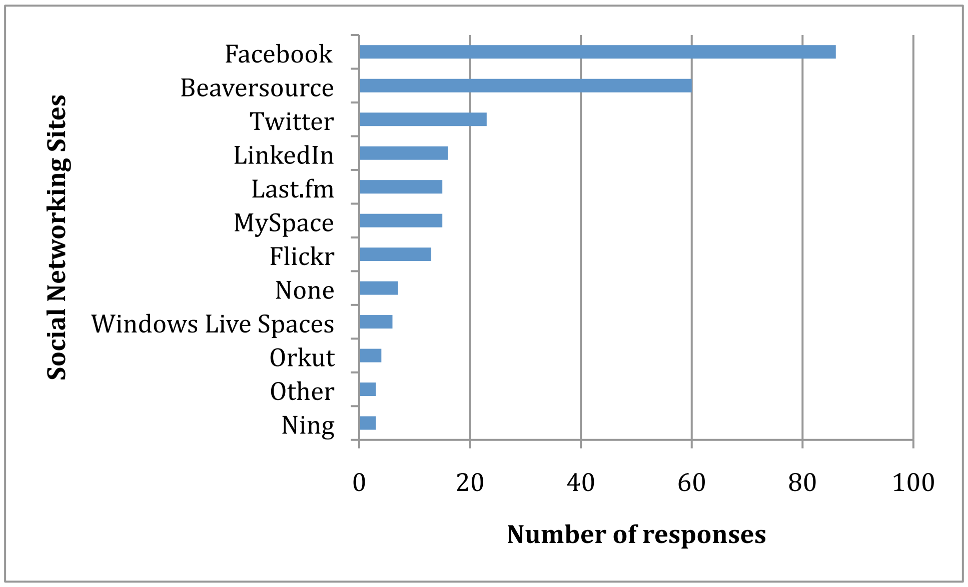


Figure 6.2 Social Networking sites used by respondents (multiple choice question)

Do you use any of the following code-hosting sites?

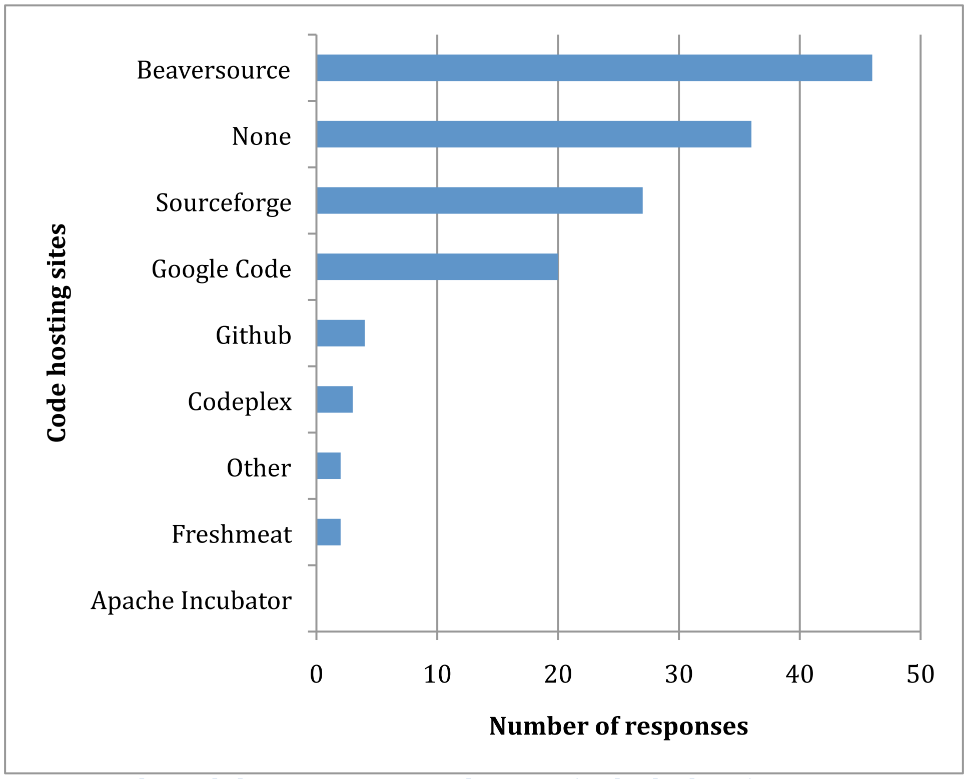


Figure 6.3 What code hosting sites respondents use (multiple choice)

Figure 6.3 shows that 35 survey respondents had not used code-hosting sites to work on projects. Comparing the data Figure 6.3 with Table 6.3 – 6.5, we can see that Beaversource is filling a need. Many students are not using or are exposed to code hosting sites, and over 40% of survey respondents used Beaversource for code hosting purposes.

6.2.1. Beaversource Experience

Beaversource Membership

Table 6.6 Beaversource membership summary

How long have you been a member of Beaversource?	Not a member – 9% < 1 term – 20% 1-2 terms – 47% 3 terms – 1 year – 17% > 1 year – 7%
How did you learn about Beaversource?	Class – 73% Email – 7% Word of mouth – 9% Search engine – 1% Barcamp – 1% Other – 9%

Forty seven percent of survey respondents were new to Beaversource with membership for 1-2 terms. The responses also showed that about 24% have been with Beaversource for 3 terms or longer. With Beaversource being a little over a year old, 7% of the users who responded to the survey were part of the beta program where only a few people had access while we were working out bugs.

Most of the survey respondents heard about Beaversource from classes. The lower level entry computer science classes, the CS senior project classes and a few more classes required students to use Beaversource. Although some people use Beaversource for classes, alternative methods had proven to be successful and we should probably focus more efforts on word of mouth and email advertising.

How often do people use Beaversource?

Table 6.7 How often people use Beaversource

How often do you use Beaversource?	Survey Respondents	People with friends	People with projects
Never	16%	5%	6%
Rarely	36%	21%	30%
Occasionally	23%	32%	27%
Several times a week	20%	32%	27%
Several times a day	5%	10%	10%

For survey respondents, the never and rarely options, which make up over 50% shows that people are not using Beaversource as often as we'd like. Only 25% of survey respondents use the site several times a week or more. Table 6.7 compares the people who have projects (1-3) and many friends (more than 6) in Beaversource. This will help us answer questions such as: Do people who have many friends use the site often? How often do people who have many projects use the site? Table 6.7 shows that there's little difference in the usage frequency of Beaversource between people with many projects and people with many friends, though the group with many friends was more likely to be on more often.

Are you a member of groups or projects in Beaversource?

Table 6.8 Group and project membership

Are you a member of groups or projects in Beaversource?	No answer – 0% Member of neither – 23% Member of a group – 24% Member of a project – 19% Member of both – 34%

Table 6.8 demonstrates that only 24% of respondents do not belong to either a project or group and 34% of respondents being a member of both.

How many projects do you belong to in Beaversource?

Table 6.9 How many projects people belong to

How many projects people belong to	**Survey respondents (N = 53)**	**Beaversource site (N = 463)**
1-3	96%	85%
4-6	0%	11%
7-9	4%	2%
10+	0%	2%

Over 45% of survey respondents chose not to answer this question, and of the remaining users, over 95% belong to at least 1-3 projects. The survey data is representative since it matches with the last column in Table 6.9, which is from the statistics of the site.

How many groups do you belong to in Beaversource?

Table 6.10 How many groups people belong to

How many groups people belong to	Survey respondents (N = 58)	Beaversource site (N = 1039)
1-3	90%	88%
4-6	7%	8%
7-9	3%	2%
10+	0%	2%

Groups in their current implementation were not being used by a lot of people. With over 40% of the users not answering this question, we can only refer to the remaining 60%. The majority of the people who answered this question, have at least 1-3 friends. This shows a shift toward social networking vs projects. Table 6.10 shows that the live site also matches the survey responses with a big percentage of people belonging to only 1-3 groups.

Why do you use Beaversource?

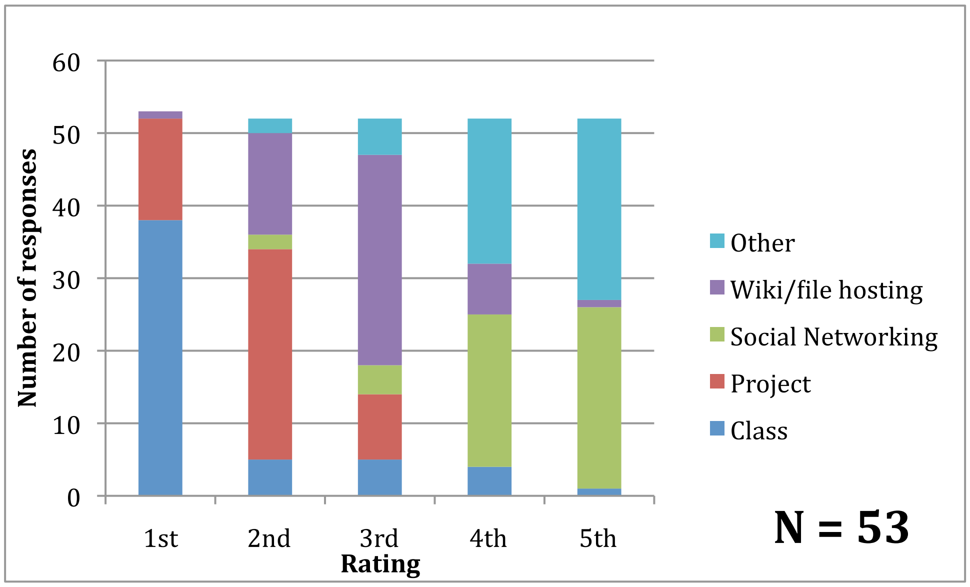


Figure 6.4 Why do survey respondents use Beaversource

People had 5 choices (class, project, social networking, wiki/file hosting and other) and they could rate them from #1 in why they chose to use Beaversource to #5. Figure 6.4 shows that the #1 reason for respondents using Beaversource is Class with more than 70% and Project with 20%. The reasons ranking the lowest for using Beaversource was Social Networking and Other.

Rate the following features

Figure 6.5 shows which features are ranked to be very useful and important in Beaversource. The lower the score, the better the ranking (1 = strongly agree and 5 = strongly disagree). Figure 6.5 shows the average score from all the submissions. The factors rated as not very important or useful are user blogs, user profiles, and project statistics. The rest of the features are very evenly spread out with their scores.

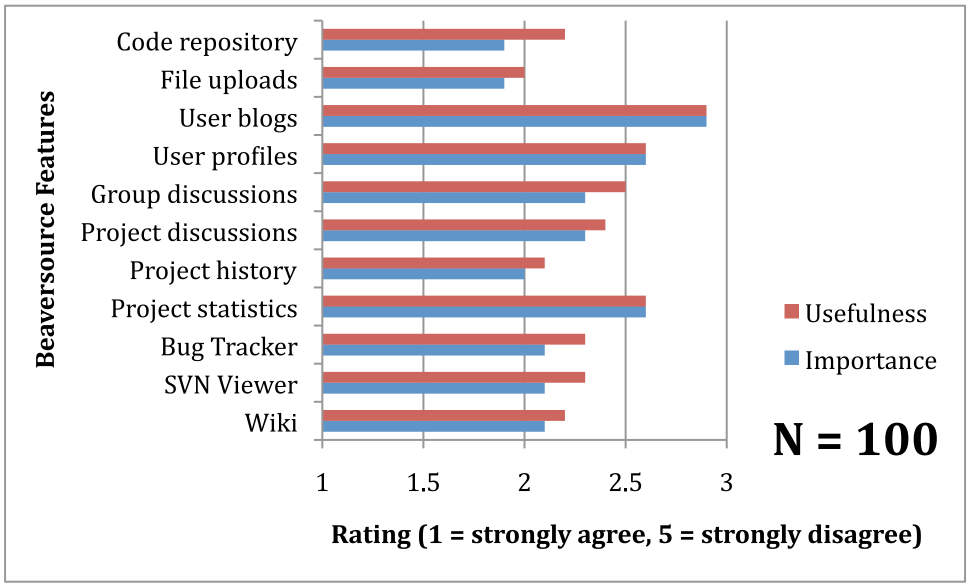


Figure 6.5 Average usefulness and importance of Beaversource Features

Do you feel that the code-hosting tools and social networking site are well integrated?

Table 6.11 Are the social and code hosting parts well integrated?

Is the social networking side well integrated with the code hosting tools?	Yes – 77% No – 21% No answer – 2%

Which features most urgently need improvement in Beaversource?

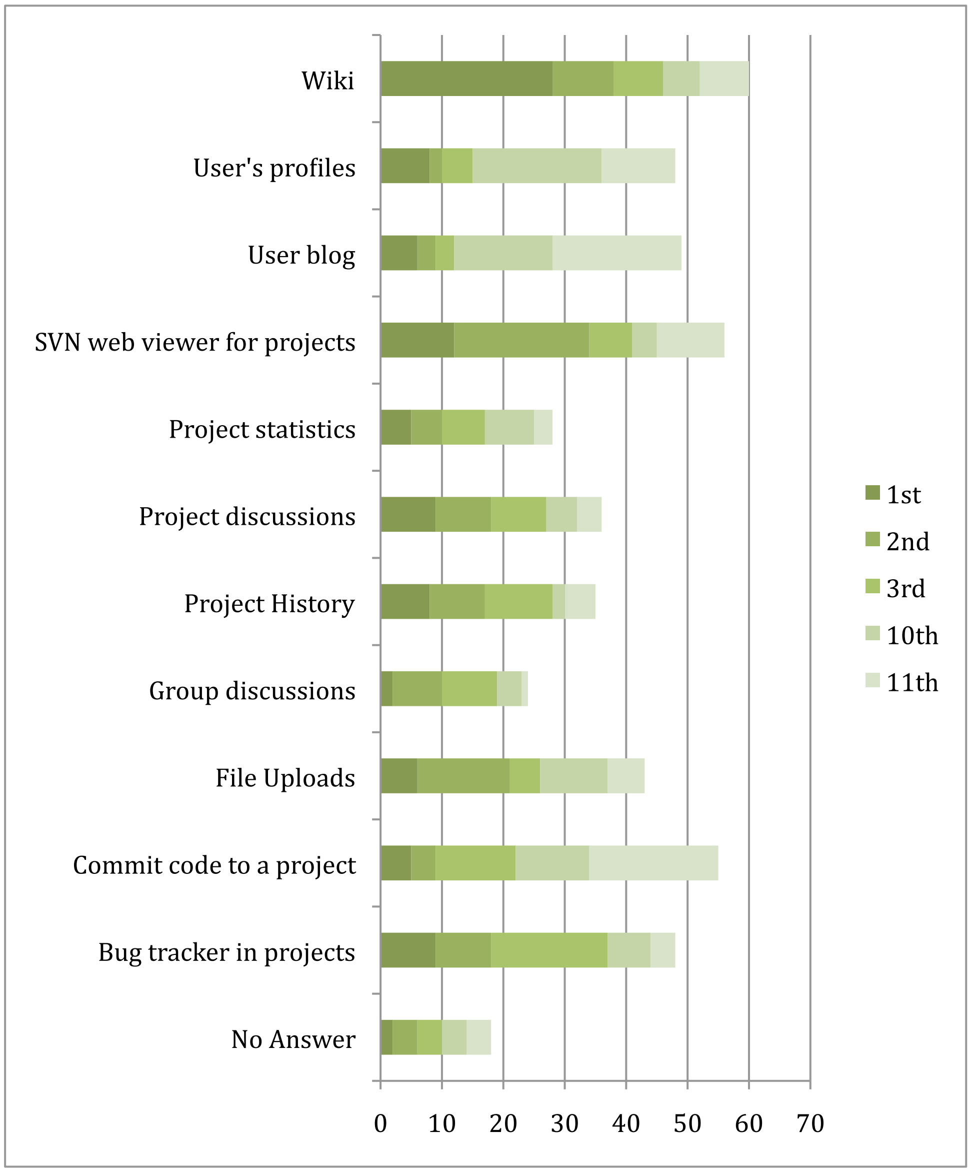


Figure 6.6 What features need improvement?

Respondents suggested that wiki is area that needs the most improvement compared to the rest. The second area that needs improvement as stated by users is SVN viewer followed by file uploads. The areas that need the least attention are the user blogs and user profiles.

Explain why social and code side are not well integrated?

Below are some of the quotes from respondents in answer to this question. An overall theme emerged:

> *"It's hard to get from a project to a person's site, hard to find different things, the search feature is not intuitive, [and] the layout is cumbersome."*

A second common answer was bugs: "I run into bugs too often." A different set of users thought that, "Social networking should not be integrated with code-hosting tools," and they were against using social networking features in Beaversource. More examples from included:

> *"Group pages for projects are confusing. To join a project you have to find the group page then join that."*

> "*The trac components feel very separate from the rest of the site. Changes on one don't well effect the changes on another."*

> *"Beaversource is very difficult to navigate, and not intuitive at all. The code repositories and social networking tools could be integrated much more smoothly if beaversource was designed better."*

6.3. Interviews

The karma system that was previously discussed was implemented as a way to motivate user interaction throughout the site. After the karma system was in place, interviews were conducted with five students to find out what they thought of the karma system, whether or not it was motivating them and how the karma system could be improved. These students were picked randomly from the set of active users. The full list of questions is available in Table 4.1.

These interviews were conducted over the summer and since the students were not on campus, the interviews were conducted remotely. Users who were interviewed did not receive any monetary payment for their help.

6.3.1. Responses

Students indicated that the wiki was a nice feature to have in Beaversource as well as the code-hosting. They found that the wiki served a documentation function. Another student pointed out that he started using Beaversource while he was a CS student, but when he changed majors he kept using Beaversource and pushed for people collaborating with him to use Beaversource's SVN functionality. This student was surprised by how few people backed up their data and didn't use code revision control.

All the students were satisfied with the code-hosting aspect of Beaversource and the functionality provided. They reported being familiar with the code-hosting tools.

Students also pointed out that browsing and searching for projects was a problem in Beaversource. They said maybe having an area where people could find recommended or suggested projects would help new people find cool projects. As far as recruiting new members to their projects, they didn't find Beaversource to be a good tool for that. Instead they tried to approach lower level classes that had lots of incoming freshmen. If somebody showed any interested at all, the groups were happy to have them. As soon as a new person started working on a group or project, the group/project members used the new member's contributions to gauge how promising that new member was.

As far as using Beaversouce as a communication tool, the students didn't find it very useful. They said they met with their group members frequently in classes and it was more efficient to use face-to-face, IRC or Facebook instead of Beaversource.

As pointed out by one student, the "Beaversource karma system, is not useful because is not about how much a student can produce, but about how much a new student wants to learn." Three students said they didn't pay attention to the karma badges. The other students said they had looked at karma badges and tried to figure out how many badges they had. They even tried to see what they could do to earn the higher level badges.

Various students pointed out that being able to grant karma badges, unique to their project members, would be fun, for example, "champions of state competition 2010." If there were more badges people could collect them.

Students also pointed out that having global forums in the site and not per group/project forums would be useful, to show off the karma badges. People interacted with other people in their group/projects and they would all know each other. The karma badge wouldn't have as much show-off worth.

One user suggested higher karma rankings for an added challenge, as well as extra badges for the testing category, which only had one badge. The users who were the most excited about karma badges were the ones that played games or checked their achievement levels in other communities. When asked for ideas on how to name badges, the interviewees didn't have any good naming strategies instead.

Another aspect mentioned by students was that potential employees or people outside of OSU might look at their profile page and see all the unique badges earned. The badges were not unique to a group/project and didn't highlight student accomplishments.

Overall, the reputation system worked to motivate users. Interviewees noted that they would like to see the karma system expanded with better icons; an increased participation in the social networking side. All users requested more flexibility in the karma system such as ability to affect each other’s karma.

7. Discussion

The reputation system put in place in Beaversource is still in beta and is a work in progress. I was experimenting with the community and wanted to see how I could best utilize karma badges to motivate people and showcase the quality of their contributions

This section covers some of the lessons I've learned during the development and feedback process of the Beaversource karma systems. I'll cover my research questions, before I address issues specific to Beaversource.

7.1. RQ1: Can a karma system boost activity in the site?

The interviews and feedback provided some early preliminary indicators, and more user data is needed to answer this research question completely. The preliminary data showed that people were looking at the karma badges and their profile page. Some of the interviewed users were motivated once they learned the scale and available badges. One of the people I interviewed wanted a progress bar to know how far away he was from achieving the last coding badge.

Reputation systems should be kept as simple as possible. These systems need to be transparent (i.e. everyone should understand what a badge means, and what needs to be done to attain one), and stay lively (i.e. stay relevant and evolve with the community) if they are to work as motivators. As reported during the literature review, users found weak areas in reputation systems and tried to exploit them. The reputation thus has to keep evolving over time: fine-tuning to account for errors, and reward as many users as possible while taking into account changes in the community.

It is feasible to construct a simple or complicated automated karma system that takes a look at the number, frequency and age of contributions, but gauging the quality of the contributions is difficult. That's where I would use user-generated karma points come into the picture. Many communities allow users to rate forum postings, or discussions with a thumbs up/down system or give karma to other users (with some limitations as to how often it can be done).

7.2. RQ2: Is it possible to reward social activity through karma badges?

Due to the low number of interviewees and with the karma system being in place during summer vacation, we do not have enough user activity and feedback to draw significant conclusions for this research question. Based on preliminary data, a few users disliked that the karma system could not be turned off. A student pointed out,

> *"How did I suddenly acquire karma badges and how do I get rid of them. As I did not place them on my page and I have no control over them I feel that anyone looking at my page sees something that is not true or anyone's business. I would greatly appreciate them removed or the means to control what gets automatically posted to my page."*

Rewarding social interactions between people through karma badges is really a delicate subject because by providing a badge, we are labeling all the users holding that badge n a way that they may not identify with. As another user pointed out, labels then become important, “men wouldn't like to be called butterfly, or chatterbox.” More user interviews are required to answer to analyze the ramifications of applying identifying labels while rewarding social activity.

The karma badges in Beaversource were meant to mark the achievements of users, and they were displayed to show how a user had progressed from one level to the next. Social interactions or friendliness was considered more of a state, people didn't like being labeled 'shy,' 'charming,' or other things that marked their personalities. Getting this type of feedback from the users involved in this project was hard. Based on early feedback, users didn't always open up or suggest better names.

A couple of things I would have done differently would be to hold a contest and get people to submit their ideas. Then we can let people vote for the badges or suggestions they liked the most. By involving the users, you could build more engagement and excitement towards the karma system.

7.3. RQ3: Can a karma system reward multiple types of activities fairly?

The standard of UI flexibility set by social sites like Facebook is high. People share a lot of personal information and they want great customizability. It was common for users to expect the features that affect everybody in a community to be either opt in or opt out. The designers of Beaversource considered the karma badges an integral part of the system, or an identifiable and core piece of information regarding a user, just like the username, full name or picture, so I chose to make it standard.

The Beaversource karma system didn't reveal any private information about the user, or their activities, just how much they contributed to projects, groups and other areas of the site. Based on the preliminary feedback received so far, I suspect that the main reason why people didn't want karma badges in their profile page was due to their dislike of the icons or label names. Next time, I might choose to let the users give each other karma badges to create more unique badges and let the users come up with names that they enjoy and feel identified with by their peers.

A lot of the information displayed in a user's profile page was configurable in the form of widgets. Users could turn on/off these features, as well as move them around on the page. The karma badges had a fixed position in the page, just like the personal bio, email, name and picture of the user. I picked a standard location in the profile page (below the profile picture) to make the badges consistent throughout the site. The position was chosen because one of the first things visitors see on a profile page was the profile picture. More interviews and data is needed to respond this question confidently, some of our findings must be taken with a small grain of salt. At least 6 more months to a year of heavy use are needed to analyze user patterns.

7.3.1. Sexist badges

The badge names were not gender neutral, and early feedback showed it as being demotivating to some users of Beaversource. As explained by a user,

> *"While I think it's a good idea to reward active users with something tangible, the badge titles seem a little sexist. They seem to be named with specific gender roles in mind. Take the coder karma badges for example. "Cowboy" and "Wizard" are generally only applied to men, thus excluding women from the coding category. In the same light, I don't think a guy would like being called a butterfly, so they wouldn't want to gain the "Social Butterfly" badge. It's a small thing, but people who are looking into joining a community can get turned off by hints of fixed gender roles (i.e. Women can't be coders and Men shouldn't be chatterboxes)."*

Finding the right set of badge names was extremely difficult, and we didn't want to pick names that would confuse new users. Having a different set of badges for male/female is not possible because the current database did not keep track of the user's gender. Also the male/female categories might not have been enough to match people's self-idea of gender. Ideally, finding a good set of gender-neutral names should not have been hard, but by having to keep things generic, the only available types of badges that I thought of were based on other disciplines:

- white belt, yellow belt, black belt
- lieutenant, sergeant, captain
- bronze, silver, gold

7.3.2. Cheating the karma system

As we found in our literature review, people have tried to cheat reputation and rating systems since they were first introduced by Amazon and EBay. It is just a matter of time before users find vulnerabilities or methods to cheat our system. Reputation systems evolve and change to meet the needs of users. It's hard to punish people who misuse the karma system to benefit themselves at the expense of other users since in most FOSS projects they can always come back using a different nickname or email account. Karma systems are not perfect and a process of continuous development has to be followed. It is usually best to reveal as little information as possible about the karma system calculations to prevent end users from learning the inner workings. This makes it harder for them to cheat the community, but makes it harder to interpret the value of a badge, or what needs to be done to attain one, lowering the system transparency and thus it's value.

7.3.3. Should karma degrade over time?

Ideally when designing a karma system, there should be a way to let new users catch up to older users in karma points. In many systems, such as Launchpad's, karma decays over time, mean that if the user is inactive for a period of time, he/she starts losing karma. On the other hand, in gamer communities, where users spend days or weeks trying to achieve high levels, it is not an accepted practice to take points away if the person stops playing the game for months. In fact, people expect their character to be at the same karma/experience level as when they left it.

In Beaversource, I chose to not decay karma over time because the students, the bulk of participants, commonly graduate, after which their accounts are deleted. Thus, the "high score list" continuously renews itself. If Beaversource were to allow alumni to participate, I would consider making karma decay over time. Conducting more user interviews to find out how users feel about karma decaying over time could help develop a better karma system.

7.4. FOSS and Karma Systems

There's little research regarding karma systems and FOSS, and we try to extrapolate some of our early findings to apply them to FOSS projects. Beaversource is geared to building communities within OSU and teach students to use code-hosting tools and the workflows used by FOSS projects. However, there are some differences. First there is huge influx of projects that are short lived, some of our users are not always happy about having to use the system, and for most this is their first experience with project hosting.

There is very little community interaction on the Beaversource mailing lists, forums and IRC channels. Most people on Beaversource only interact with peers in their projects or friends in class. Also people who are experienced with FOSS projects usually prefer to roll out their own code-hosting set of tools or use an externally hosted site such as GitHub or Sourceforge.

In FOSS communities, people are remembered for their contributions, and developers come and go from projects. Developers or community members who take breaks for weeks or months come back to a community that still remembers their names thanks to their contributions, such as patches, mailing list interactions and forum posts. Karma systems are supposed to mimic earned reputation, but some of the earned reputation always stays with the person. For example, Linus Torvalds, gets lots of geek points for creating and getting the Git project started even though he is no longer actively coding or actively participating in that community.

A simple approach could be taken to design reputation systems in FOSS communities. Instead of having one set of karma badges or a comprehensive karma system that looked at different types of activities to grant badges, it might be more useful to have different karma systems, but one unified display of badges. This could prove to be more resistant to attacks because if one karma systems is compromised the rest of them would not be. For example, the forum software could take care of mining its own data and use its metrics to generate badges; the IRC channel could use an IRC bot to count the number of times a user posted information or tried to help other users, then it could calculate a set of badges; a separate reputation system could take care of looking through the code repository and use a bug tracker to find code contributions.

This is the area that makes reputation systems unfair and hard to design for FOSS projects. Giving points for different actions through an automated system is difficult, especially when trying to compare different things such as fix bug to provide end user support. A user may have contributed a small amount of time, but the bug fix/documentation/help provided may have impacted the community drastically, and thus time spent is not always a good metric. A new user might have spent hours or days working on a contribution, even if it was small, what should be rewarded is the effort and desire to work, not the size or impact of the contribution. On the other hand, core developers familiar with the codebase, bugs and project spend a sizeable amount of time each week to find bugs and add new features, yet they shouldn't be achieving the next badge/reputation level as fast as new members.

Trying to figure out how much documentation is “worth” vs. forum support vs. coding vs. lurking in irc is a common dilemma. As discussed during the literature review, most FOSS projects ignore most of the activities and focus on providing karma pointes based on just one type of activity. It's not fair to the community to try to place value or compare different types of contribution. Also the reputation systems should be more about trying to promote exemplary behavior instead of rewarding a user just because an action was performed.

7.5. Beaversource's Karma Implementation

After the surveys were collected, we found that there were few users having a reason to use Beaversource for the social elements. Most were already interacting with friends and family using Facebook. Having to enter their information and use a second site was not appealing based on the preliminary feedback (answers to the survey). People pointed out that the site needed something to motivate people and differentiate from the other sites.

The use of the project hosting tools seems to be successful. People had projects and they found it useful, but frustrating for class usage. Some people asked questions that were due to their lack of experience and exposure to code-hosting tools. It was the goal of the main developer to keep those users happy while we were trying to recruit more users.

7.5.1. Badge Names

As was discussed previously, the reputation level names were difficult for me to come up. I changed the names many times, but at the end I couldn't find a good answer. Neutral names that please every user and are accessible to everybody are hard to design. By using more generic titles for the badges, I lost the uniqueness factor of the badges. I tried asking users for new names, but got not feedback.

7.5.2. No Opt Out

Beaversource badges were a requirement just like the name, email and username of a person. This became a sore area for some users. Historically users who didn't like the social side of Beaversource were able to just ignore it and not update their profiles. The karma system used their project contributions and other history data to calculate karma points and displayed the badges in the users' profiles. This caused a major shift for some users. Some of them felt as though the karma system was forced on them and they didn't want the information in their profiles.

7.5.3. No good place to showcase badges.

The fact that there was a bunch of bugs that people kept running into was very demotivating, as shown by early feedback. People who had worked hard to earn karma badges only had their badges displayed in their profiles and the user listing pages. There is no single place where Beaversource users could come and idle, hang out or interact with each other outside of projects (such as a lounge). If there had been such a place showcasing the user's badges, when they made a forum post or interacted with each other would have increased the visibility of their karma badges.

Due to some of the differences in how Trac (a code-hosting tool) displayed user information, I couldn't display karma badges alongside the username. The one place where a lot of users spent their time didn't even show their badges. Even if the badges had been displayed, users would only have gotten to see the badges of peers outside their projects. They wouldn't have seen a variety of different users with a wide range of badges. Instead they saw the same users they're used to with the same set of badges. We are still in the early development stages of the karma system to make sound conclusions or statements regarding the system.

7.5.4. Rewarding Quantity instead of Quality

My karma badge system was automated which and had one major flaw in that it counted the number of times various actions have been performed in the site. Based on that piece of information various badges were awarded to users. There was a problem with this approach. We are not rewarding what we want. By just rewarding actions, I didn’t take into account the quality of user's contributions. I learned that two badges can mean two completely different things if one user just spent time entering simple repetitive tickets into the system or one line commits that don't add anything to a project and the other user made huge contributions.

By rewarding quantity, I was leaving the system vulnerable to attacks. A programmer could easily have written a program that would have submitted and updated a bunch of tickets, wiki pages and commits. This would then have earned the user the highest-level karma badges in the scale. Also, by not rewarding quality, I was unable to showcase or demonstrate why one user or set of behaviors is better than another one.

8. Conclusion

A continuous influx of new active members is what keeps FOSS communities afloat. As end users move from being lurkers to core developers, finding a method to measure the number of community members, and motivate users to contribute more to the community, is a challenge faced by all FOSS projects, whether small or large.

At Oregon State University, Beaversource provides code-hosting services to students, staff and faculty, with a mix of social networking features. The social networking features were meant to help users find interesting projects and people to collaborate with. I conducted a survey to learn about user demographics, and their experience with Beaversource, especially the social networking features.

As an attempt to motivate users and reward active members, a reputation system was developed. Unlike reputation systems in other FOSS projects that focus only on one community activity, I designed the Beaversource system to reward a wide range of activities on the site. The system went through several iterations and it was made available to all members.

With this being the first attempt at providing a reputation system for Beaversource, we learned a lot. Preliminary findings showed that the system did motivate some users, but there was a group of users who wanted more control of their reputation display. I also realized that due to the limitations of our automated karma system, the reputation system was rewarding the actions and not the quality of the actions performed by the users. Even though there was a karma system in place, the end users didn't interact with people outside of their projects or close-knit group of friends, so there was very little chance for users to showcase their badges. The preliminary feedback from interviews and survey showed that the reputation system was not effective, because of limitations of the current Beaversource platform with bugs and lack of global groups or lounge areas.

It's my hope that future researchers can learn from our experiences with designing an all-encompassing reputation system: Users expect a high level of customizability features from karma systems. Mixing social networking features with code-hosting features makes it hard to create reputation labels that reward social interactions. People take labels regarding their social networking behaviors very personal.

Future steps for this research are to focus on fixing some of the bugs in Beaversource that limit the use of karma badges. It will be important to delve deeper into the question of how to reward social behavior, such as adding friends and interacting with them. Analyzing more of the benefits of mixing social networking features with code-hosting tools may lead to a new breed of code-hosting sites that can empower future FOSS communities by connecting people and ideas together.

9. Bibliography

21 Days, 15 Hours, 26 Minutes and 2 Seconds. (n.d.). Retrieved from http://habitatchronicles.com/2007/09/21-days-15-hours-26-minutes-and-2-seconds/

ACM. (2008). CS2008 Curriculum Update: The Computing Curricula Computer Science Volume is complete and approved.

Adler, B. T., & de Alfaro, L. (2007). A content-driven reputation system for the wikipedia. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07 (pp. 261–270). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1242572.1242608

Apache Incubator. (n.d.). *http://incubator.apache.org/*.

Au. A E Arenas (STFC), A. B. A. (. (2008). Reputation Management in Grid-Based Virtual Organisations. In *International Conference on Security and Cryptography*.

Axelrod, R., & Hamilton, W. D. (1981). The evolution of cooperation. *Science*, *211*(4489), 1390-1396. doi:10.1126/science.7466396

Beware Geeks Bearing Gifts. (n.d.). Retrieved from http://www.psychologytoday.com/blog/brainstorm/200803/beware-geeks-bearing-gifts

Bird, C., Gourley, A., Devanbu, P., Gertz, M., & Swaminathan, A. (2006). Mining email social networks. In *Proceedings of the 2006 international workshop on Mining software repositories*, MSR '06 (pp. 137–143). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1137983.1138016

Boy Scouts of America - Introduction to Merit Badges. (2010). Retrieved from http://www.scouting.org/scoutsource/BoyScouts/AdvancementandAwards/MeritBadges.aspx

Burke, M., & Kraut, R. (2008). Mind your Ps and Qs: the impact of politeness and rudeness in online communities. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 281–284). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460609

Cathedral and the Bazaar. (2010). Retrieved from http://catb.org/esr/writings/homesteading/

Chatterjee, K., de Alfaro, L., & Pye, I. (2008). Robust content-driven reputation. In *Proceedings of the 1st ACM workshop on Workshop on AISec*, AISec '08 (pp. 33–42). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1456377.1456387

Clarkson Open Source Institute. (2010) . Retrieved from http://cosi.clarkson.edu/

Clarkson University wins first TuxMasters Invitational. (2010). . Retrieved from http://www.linux.com/archive/articles/47881

Couple updates... (2010). Retrieved from http://about.digg.com/blog/couple-updates…

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, *10*(2).

Cummings, J. N., & Kiesler, S. (2008). Who collaborates successfully?: prior experience reduces collaboration barriers in distributed interdisciplinary research. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 437–446). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460633

Deiml-Seibt, T., Pschetz, L., & M"uller, B. (2009). A conversational model to display user activity. In *Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology*, BCS-HCI '09 (p. 6). Swinton, UK, UK: British Computer Society.

Dugan, C., Geyer, W., Muller, M., DiMicco, J., Brownholtz, B., & Millen, D. R. (2008). It's all 'about you': diversity in online profiles. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 703–706). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460672

Employers: Look to gaming to motivate staff. (n.d.). . Retrieved from http://www.itnews.com.au/News/169862,employers-look-to-gaming-to-motivate-staff.aspx

Feller, J. (2010). Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering.

Fitzpatrick, G., Marshall, P., & Phillips, A. (2006). CVS integration with notification and chat: lightweight software team collaboration. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06 (pp. 49–58). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1180875.1180884

FOSS. (2010). *Wikipedia*. Retrieved September 6, 2010, from http://en.wikipedia.org/wiki/Free_and_open_source_software#FOSS

Foster, D. (2010, February). *Chat with Dawn Foster*.

Foursquare. (2010). Retrieved from http://foursquare.com/learn_more

Friedman, E. J., Friedman, E. J., & Resnick, P. (2000). The Social Cost of Cheap Pseudonyms. *JOURNAL OF ECONOMICS AND MANAGEMENT STRATEGY*, *10*, 173–199. doi:10.1.1.30.6376

Fu, W. (2008). The microstructures of social tagging: a rational model. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 229–238). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460600

Game Mechanics for Interaction Design: An Interview with Amy Jo Kim. (n.d.). . Retrieved from http://bokardo.com/archives/game-mechanics-for-interaction-design-an-interview-with-amy-jo-kim/

Good Game Mechanics In Your Web App Are Good For Your Users. (n.d.). . Retrieved from http://blog.meatinthesky.com/good-game-mechanics-in-your-web-app-are-good

Google and Open Source. (n.d.). Retrieved from http://code.google.com/opensource/

Google Project Hosting. (2010). Retrieved from http://code.google.com/projecthosting/

Google Summer Of Code. (2010). Retrieved from http://socghop.appspot.com/document/show/gsoc_program/google/gsoc2010/faqs

Gourley, B. (2009). Open Source Software and Cyber Defense.

Halloran T. J., S. W. L. (2002). *High Quality and Open Source Software Practices*.

Halverson, C. A., Ellis, J. B., Danis, C., & Kellogg, W. A. (2006). Designing task visualizations to support the coordination of work in software development. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06 (pp. 39–48). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1180875.1180883

Hancock, J. T., Toma, C. L., & Fenner, K. (2008). I know something you don't: the use of asymmetric personal information for interpersonal advantage. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 413–416). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460629

Harper, F. M., Frankowski, D., Drenner, S., Ren, Y., Kiesler, S., Terveen, L., Kraut, R., et al. (2007). Talk amongst yourselves: inviting users to participate in online conversations. In *Proceedings of the 12th international conference on Intelligent user interfaces*, IUI '07 (pp. 62–71). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1216295.1216313

Harvard Forge. (2010). . Retrieved from http://forge.abcd.harvard.edu/gf/

Hoffman, K., Zage, D., & Nita-Rotaru, C. (2009). A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, *42*, 1:1–1:31. doi:http://doi.acm.org/10.1145/1592451.1592452

HomeRun. (2010). . Retrieved from http://homerun.com/how-it-works

Hossain, L., & Zhou, D. (2008). Measuring OSS quality trough centrality. In *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, CHASE '08 (pp. 65–68). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1370114.1370131

How does MediaWiki Work? (n.d.). . Retrieved from http://www.mediawiki.org/wiki/How_does_MediaWiki_work%3F

I love my chicken wire mommy. (n.d.). Retrieved from http://xoxco.com/clickable/i-love-my-chicken-wire-mommy

Ingram, M. (n.d.). Huffington Post Does a Foursquare, Offers Readers Badges for Behavior. Retrieved from http://www.businessweek.com/technology/content/apr2010/tc20100429_746797.htm

Irani, L. C., Hayes, G. R., & Dourish, P. (2008). Situated practices of looking: visual practice in an online world. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 187–196). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460592

IRC. (2010). Retrieved from http://en.wikipedia.org/wiki/Internet_Relay_Chat_bot

Kernel Janitors. (2010). Retrieved from http://kernelnewbies.org/KernelJanitors

Kernel Newbies. (2010a). Retrieved from http://kernelnewbies.org/Community

Kernel Newbies. (2010b). Retrieved from http://kernelnewbies.org/KernelProjects

Kerschbaum, F. (2009). A verifiable, centralized, coercion-free reputation system. In *WPES '09: Proceedings of the 8th ACM workshop on Privacy in the electronic society* (pp. 61–70). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1655188.1655197

Ko, A. J., & Chilana, P. K. (2010). How power users help and hinder open bug reporting. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems* (pp. 1665–1674). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1753326.1753576

Ko, E. (2010, February). *Chat with Ellen Ko*.

Lampe, C., Ellison, N. B., & Steinfield, C. (2008). Changes in use and perception of facebook. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 721–730). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460675

Lampe, C., & Resnick, P. (2004). Slash(dot) and burn: distributed moderation in a large online conversation space. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04 (pp. 543–550). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/985692.985761

Launchpad Karma. (2010). Retrieved from https://help.launchpad.net/YourAccount/Karma

Launchpad Mailing lists. (n.d.). Retrieved from https://help.launchpad.net/Teams/MailingLists

Lee, K. J. (2006). What goes around comes around: an analysis of del.icio.us as social space. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06 (pp. 191–194). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1180875.1180905

Luther, K., & Bruckman, A. (2008). Leadership in online creative collaboration. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08 (pp. 343–352). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1460563.1460619

Metrics for Healthy Communities. (2010). Retrieved from http://www.horsepigcow.com/2007/10/metrics-for-healthy-communities/

Mozilla Computer Science. (2010). Retrieved from https://wiki.mozilla.org/Education/ComputerScience

Mozilla Education. (2010). Retrieved from https://wiki.mozilla.org/Education/Overview

Munga, N., Fogwill, T., & Williams, Q. (2009). The adoption of open source software in business models: a Red Hat and IBM case study. In *SAICSIT '09: Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists* (pp. 112–121). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1632149.1632165

MyBB Feature Tour. (2010). Retrieved from http://www.mybb.com/features

Ohira, M., Ohsugi, N., Ohoka, T., & Matsumoto, K. (2005). Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. *SIGSOFT Softw. Eng. Notes*, *30*, 1–5. doi:http://doi.acm.org/10.1145/1082983.1083163

Ohloh. (2010). Retrieved from www.ohloh.net/

Open Source As Programming Exp. for College Students. (2010). . Retrieved from http://ask.slashdot.org/article.pl?sid=02/02/26/198253&mode=nested&tid=156

Open Source Initiative. (2010, OSI). Retrieved from http://www.opensource.org/history

Open Source Software. (2010). Retrieved from http://en.wikipedia.org/wiki/Open-source_software

OSL University College Dublin, Ireland. (2010). . Retrieved from http://osl.ucd.ie/

OSU OSEL. (2010). Retrieved from http://osel.oregonstate.edu/index.php?title=Main_Page

OSU OSL. (2010). Retrieved from http://osuosl.org/about-osuosl

Pavlou, P. A., Pavlou, P. A., & Gefen, D. (n.d.). Psychological Contract Violation in Online Marketplaces. *ANTECEDENTS, CONSEQUENCES, AND MODERATING ROLE," INFORMATION SYSTEMS RESEARCH*, *16*, 372–399. doi:10.1.1.87.24

Pink, D. (n.d.). Daniel Pink on Motivation. Retrieved from http://www.youtube.com/watch?v=_mG-hhWL_ug

Planet. (n.d.). Retrieved from http://www.planetplanet.org/

Porter, J. (2010). Is Harriet Klausner for real? Retrieved from http://bokardo.com/archives/is-harriet-klausner-for-real/

Problems and Cheaters Curb Stomp Emergence Day Problems and Cheaters Curb Stomp Emergence Day dfProblems and Cheaters Curb Stomp Emergence Day Problems and Cheaters Curb Stomp Emergence Day Problems and Cheaters Curb Stomp Emergence Day Problems and Cheaters Curb Stomp Emergence Day. (n.d.). . Retrieved from http://www.2old2play.com/News/Problems_and_Cheaters_Curb_Stomp_Emergence_Day#7058

Resnick, P., Kuwabara, K., Zeckhauser, R., & Friedman, E. (2000). Reputation systems. *Commun. ACM*, *43*, 45–48. doi:http://doi.acm.org/10.1145/355112.355122

REVOLUTION OS. (2001, Revolution OS). Retrieved from http://www.revolution-os.com/

Riehle, D. (2006). How and why Wikipedia works: an interview with Angela Beesley, Elisabeth Bauer, and Kizu Naoko. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis* (pp. 3–8). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1149453.1149456

Robbins, J. E., & Robbins, J. E. ". (2002). Adopting OSS Methods by Adopting OSS Tools. doi:10.1.1.11.1951

Sen, S., Lam, S. K., Rashid, A. M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, F. M., et al. (2006). tagging, communities, vocabulary, evolution. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06 (pp. 181–190). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1180875.1180904

Sibisi, S., Jensen, M., Machanick, P., & Blake, E. (2004, July). Free/Libre & Open Source Software and Open Standards in South Africa. Retrieved from http://www.naci.org.za/pdfs/floss_v2_6_9.pdf

Social media games: Badges or badgering? (2010). . Retrieved from http://www.cnn.com/2010/TECH/05/04/cnet.foursquare.badges/index.html?hpt=Sbin

SourceForge. (2010). Retrieved from http://sourceforge.net/develop

de Souza, C., Froehlich, J., & Dourish, P. (2005). Seeking the source: software source code as a social and technical artifact. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, GROUP '05. New York, NY, USA: ACM.

Spolsky, J. (2010). Painless Bug Tracking. Retrieved from http://www.joelonsoftware.com/articles/fog0000000029.html

StackOverflow. (2010). Retrieved from http://stackoverflow.com/faq

StackOverflow Badges. (2010). Retrieved from http://stackoverflow.com/badges

Stanford Open Source. (2010). Retrieved from https://opensource.stanford.edu/

Su, H., Jodis, S., & Zhang, H. (2007). Providing an integrated software development environment for undergraduate software engineering courses. *J. Comput. Small Coll.*, *23*, 143–149.

Teaching Open Source. (2010). Retrieved from http://teachingopensource.org/index.php/Main_Page

Tigris. (2010). Retrieved from http://www.tigris.org/

Trac Wiki. (n.d.). Retrieved from http://trac.edgewall.org/wiki/TracWiki

Tracking top diggers. (2010). Retrieved from http://www.chrisfinke.com/2008/05/23/tracking-the-top-diggers/

Unable to parse string as BibTeX. (n.d.).

Walsh, K., & Sirer, E. G. (2005). Fighting peer-to-peer SPAM and decoys with object reputation. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems* (pp. 138–143). New York, NY, USA: ACM. doi:http://doi.acm.org/10.1145/1080192.1080204

Yahoo Answers. (2010). Retrieved from http://answers.yahoo.com/info/scoring_system

YDN - Reputation Patterns. (n.d.). Retrieved from http://developer.yahoo.com/ypatterns/social/people/reputation/

Ye, Y., & Kishida, K. (2003). Toward an understanding of the motivation Open Source Software developers. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03 (pp. 419–429). Washington, DC, USA: IEEE Computer Society. Retrieved from http://portal.acm.org/citation.cfm?id=776816.776867